



UNIVERSITY OF  
LIVERPOOL

2017/18

**Student Name:** David Owens

**Student ID:**

**Project Title:** COMP39X  
Web Scraping

**Supervisor:**

**DEPARTMENT OF  
COMPUTER SCIENCE**

## **Contents**

1.0: Abstract.....	3
2.0: Introduction .....	4
3.0: Background .....	5
3.1: Existing Solutions .....	6
3.2: Information Gathering .....	6
3.3: Statement of Deliverables .....	7
3.4: Essential Features .....	7
3.5: Desirable Features .....	7
4.0: Data Required .....	7
4.1: Ethical Use of Data .....	9
5.0: Design.....	10
5.1: Summary of Proposal.....	10
5.2: Description of System .....	11
5.3: Data Dictionaries.....	12
5.4: Logical Table Design.....	13
5.5: Physical Table Design .....	13
5.6: System Boundary Diagram.....	14
5.7: Data Flow Diagrams .....	14
5.7.1: Amazon kindle top 100 Scrape .....	14
5.7.2: Specific Book Search for Additional Information.....	15
5.8: Files Produced.....	15
5.8.1: Top 100 Scrape.....	15
5.8.2: Specific Book Scrape .....	16
5.8.3 Update Book .....	18
5.8.4 Delete Book.....	19
5.8.5 Add Book.....	19
5.9: Use Case Diagrams.....	21
5.9.1: Old Use Case Diagram .....	21
5.9.2: New Use Case Diagram .....	22
5.10: Use Case Tables.....	23
5.11: Evaluation Design.....	27
5.12: Process Map.....	29
5.13: UI Design .....	30
5.14: Updated Gantt Chart for Implementation Phase .....	32
6.0: Realisation.....	33

6.1: Prototype 1 .....	33
6.2: Prototype 2 .....	35
6.3: Prototype 3 .....	38
6.3.1: addBook .....	39
6.3.2: deleteBook .....	40
6.3.3: specificBookTop100Scrape .....	40
6.3.4: updateBook.....	40
6.4: GUI .....	41
6.5: HTML Change .....	42
6.6: Prototype 3.5.0 (soupTest3Selenium) .....	43
6.7: Prototype 3.5.1 (soupSelenium) .....	43
6.8: Prototype 4 .....	44
6.9: Demonstration Build (Final Build) .....	44
7.0: Evaluation .....	45
7.1: Essential Features .....	45
7.2: Desirable Features .....	46
7.3: Evaluation from Design.....	47
7.4: Results/Outcomes.....	47
7.5: Results.....	48
7.6: Strengths and Weaknesses .....	50
7.6.1: Strengths .....	50
7.6.2: Weaknesses .....	50
8.0: Learning Points.....	51
9.0: Code of Practice .....	52
10.0: Bibliography .....	54

## **1.0: Abstract**

Web scraping refers to techniques for extracting information directly from web pages, generally referring to a process performed by an automated software. Data is collected directly from webpages and stored in some other location, normally a database or spreadsheet.

There are many different techniques to scrape information from web pages, ranging from relatively simple programs using regular expressions to find the data required up to much more complex systems utilising vertical aggregation or machine learning to perform web scraping on a much larger scale.

This project involved the creation of appropriate tools for web scraping, which were then used to analyse data contained on Amazon pages. More specifically, the project allowed the Amazon Kindle top 100 list to be scraped over a period of time and the results of scrapes to be stored for later analysis.

All tasks performed by this web scraping tool could, in theory, be done by a human extracting the data directly from the webpages and inserting said data into a database manually. This can also be said of all web scraping tools. However, with potentially over a hundred new database records being created each hour, this process is highly inefficient for a human to perform. The software created by this project sought to automate this repetitive procedure of data extraction and storage and provide some analysis on the data retrieved.

Perhaps the most important thing to note about this project is that the final system produced may no longer work. Nearing the end of the project the HTML of the pages being scraped was altered and after a short time reverted. It is unknown whether the new or old webpages are currently being utilised. As such this project has been completed using a potentially outdated version of the live webpages, and the final piece of software created may need to be updated in order for it to function as intended.

Despite this the project was overall a success and a system which allowed for the scrape and storage of data from the Amazon Kindle top 100 was produced. The program can be used to build up a database of book placements in the Amazon Kindle top 100 over time, and each book can be inspected further to see how a book's data has changed over time.

## **2.0: Introduction**

Web scraping refers to techniques for extracting information directly from web pages. This project involved the creation of appropriate tools for this, which were then used to analyse data contained on Amazon pages, namely the Amazon Kindle top 100.

The aim of the project was to create a web scraping program which would scrape some data from a set of webpages and store the results for later analysis.

My proposed solution to the project was a program, with a GUI written in Visual Basic, a database system implemented in MySQL, and main code written in Python, which allowed the user to; scrape the Amazon Kindle top 100; view the results of scans; view the data stored on the books that have been scanned; add and remove books from the database; and view how the data stored on any book in the database has changed over time, for example how the rank of the book has changed over the series of scans it had been found in.

The project came with numerous risks. I had never programmed using Python so the project was likely to take longer than if I was using a language I was more familiar with, potentially resulting in delays. This was considered in the estimation time for each section of the project involving Python coding and was remedied through the use of online tutorials and assistance from some other members of the University.

If the website being scraped went down at any point the program will simply not work. This was entirely out of the control of the project but should not be too much of a problem as Amazon is one of the largest websites on the internet and as such should not be down for too long if indeed it does go down. A similar problem to this did actually occur during the project and will be discussed in section 6.5 of this document.

I also consider UI design to be one of my weaknesses so creating a suitable GUI for the project was a challenge.

Until a week before the demonstration was due to be given, the solution to the project was effective. Data was able to be scraped from the Amazon Kindle top 100 without error, a functional GUI was developed to provide basic analysis on the data gained over a period of time, and the system allowed the user to perform scans and manipulate the database.

Unfortunately for the project Amazon changed the layout of the Kindle top 100 so that, instead of being 5 pages long with 20 books on each page, it now consisted of 2 pages with 50 books on each. As a result of this change the system produced for the demonstration, and indeed the final build of the project, would not work on the live version of the Kindle top 100 at the time and as a result the demonstration was performed on an offline version of the webpages.

An alternative, working build was created specifically for the updated version of the top 100. Because of the short time frame, and other complications that arose (discussed in more detail in the Realisation section of this document), a GUI was not produced for this new

version of the system. The newest version of the system was also not split into multiple files, and instructions on how to run this version of the project can be found in the code itself.

Regardless of outside factors, the project can still be considered a success since a tool was developed which allowed the scraping of data from a set of Amazon pages and allowed analysis to be performed.

Perhaps not as much analysis as intended has been performed on the data built up by the current version of the program, which currently graphically shows how a book's data has changed over time. The system does presently provide an additional example of some analysis which can be performed on the data, such as the date of the scan each book was first and last found in, and the average rank of the book over all the scans it has been found in. These are just basic examples of what analysis can be performed on the data gathered by the system and the project could easily be extended in the future to provide much more meaningful analysis; from something as simple as multiple graphs on the GUI to allow the user to see price spikes and any potential effects on other parts of the data, or seeing if any particular author or publisher appears much more often than any others, or something else entirely more specific to the client's desire.

To ensure the ongoing success of the project, if it were to continue, the program would need to be updated whenever the creator of the website being scraped decides to update their HTML. This shouldn't happen too often, as the Kindle top 100 had been separated into 5 pages as far back as 10<sup>th</sup> March 2013 [1] and it was ill-fated that the change occurred during the development of the system.

### **3.0: Background**

Web scraping refers to techniques for extracting information directly from web pages, generally referring to a process performed by an automated software. Data is collected

directly from webpages and stored in some other location, normally a database or spreadsheet.

This process involves first fetching the web page in question, meaning the page is downloaded, and then extracting any data that is required from it, for example by searching through the HTML code of the webpage the exact location of the required information can be found and extracted. This data can then be used later for many purposes, such as analysis or simply as an archive.

More in depth descriptions of web scraping are available elsewhere [2], but for the purposes of this project the definition above will suffice.

### **3.1: Existing Solutions**

There are a large number of web scraping programs freely available on the internet. One such example found during research is a data scraper available as an extension to the Google Chrome web browser which “automatically locates and extracts data from web pages” [3]. Further instructions on how to use the system and what it can do can be found on their website. This system was studied to get a greater idea of how web scraping can be performed- their video [4] containing instructions on how to use the system gave a better idea of how data can be extracted from the HTML of a webpage. Additionally, this system allows data to be exported to an Excel or CSV file, which contributed towards the idea of using a graph as part of the GUI in this project.

WebHarvy [5] is another web scraper looked at during the research phase of this project. A video [6] was studied in which multiple Amazon pages themselves were scraped and data successfully extracted. This resulted in the project being known to be feasible and gave a good starting point to deciding what data would be extracted from the website.

Both of the above two solutions are more generalised than the system created during this project. They can both be utilised to perform the extraction part of this project but are far more suited for larger scraping projects with several different webpages, each with differing designs.

A more scientific system used to scrape climate data [7] was also studied to gain a better understanding of web scraping and further cemented the idea of using a graph in the GUI of this project.

### **3.2: Information Gathering**

Multiple websites were used to get a better understanding of how web scraping can be performed specifically with the Python programming language. After some research it was decided that the BeautifulSoup module would be suitable for this project. The official documentation [8] was helpful when beginning to use the module, and also throughout the project so as to get a better understanding of what the module was capable of.

Another source used when researching the usefulness of the BeautifulSoup module, and web scraping in general, can be found here [9]. This gave an excellent tutorial on how a

beginner could start to create a web scraping program using the tools that were chosen to be used in the project. Another similar blog post [10] was used in the same way.

### **3.3: Statement of Deliverables**

The project required; design documentation and a presentation to accompany it, to be delivered by Thursday 16 November 2017; an interim progress report, to be delivered Friday 16 February 2018; a presentation of the project, to be delivered Thursday 12 April 2018; and a final project report to be delivered Thursday 10 May 2018.

### **3.4: Essential Features**

- Allow the user to scrape the Amazon kindle top 100 and extract details
- Allow the user to scrape the Amazon top 100 to find a specific book, given its URL
- Store basic information, URL and current rank of each item in a database
- Show the user an overview of a specific item such as; how long it has been in the top 100; how long it has been at its current rank; the item's highest historical rank; and dates/times of when data was retrieved.
- Provide a simple interface to allow the user to perform the above

### **3.5: Desirable Features**

- Construct a graph, shown to the user upon request, which shows how the ranking of a book has changed over time
- Allow side by side comparison of at least 2 books
- Allow the user to scan both the free and paid top 100
- Automate the process of scraping each hour for as long as the user is running the program

## **4.0: Data Required**

The data required for the system was scraped from the Amazon Kindle top 100 pages, which comprised of 5 web pages, and additionally from Amazon webpages for specific books. Each piece of data required can be extracted directly from the HTML of the webpage.



The 5 pages of the Amazon Kindle top 100 can be found, as of the time of the creation of this document, at the following URLs:

1. [https://www.amazon.co.uk/Best-Sellers-Kindle-Store/zgbs/digital-text/ref=zg\\_bs\\_pg\\_1?\\_encoding=UTF8&pg=1](https://www.amazon.co.uk/Best-Sellers-Kindle-Store/zgbs/digital-text/ref=zg_bs_pg_1?_encoding=UTF8&pg=1)
2. [https://www.amazon.co.uk/Best-Sellers-Kindle-Store/zgbs/digital-text/ref=zg\\_bs\\_pg\\_2?\\_encoding=UTF8&pg=2](https://www.amazon.co.uk/Best-Sellers-Kindle-Store/zgbs/digital-text/ref=zg_bs_pg_2?_encoding=UTF8&pg=2)
3. [https://www.amazon.co.uk/Best-Sellers-Kindle-Store/zgbs/digital-text/ref=zg\\_bs\\_pg\\_3?\\_encoding=UTF8&pg=3](https://www.amazon.co.uk/Best-Sellers-Kindle-Store/zgbs/digital-text/ref=zg_bs_pg_3?_encoding=UTF8&pg=3)
4. [https://www.amazon.co.uk/Best-Sellers-Kindle-Store/zgbs/digital-text/ref=zg\\_bs\\_pg\\_4?\\_encoding=UTF8&pg=4](https://www.amazon.co.uk/Best-Sellers-Kindle-Store/zgbs/digital-text/ref=zg_bs_pg_4?_encoding=UTF8&pg=4)
5. [https://www.amazon.co.uk/Best-Sellers-Kindle-Store/zgbs/digital-text/ref=zg\\_bs\\_pg\\_5?\\_encoding=UTF8&pg=5](https://www.amazon.co.uk/Best-Sellers-Kindle-Store/zgbs/digital-text/ref=zg_bs_pg_5?_encoding=UTF8&pg=5)

The data scraped from these 5 pages is as follows:

Data	Description
ASIN	The ASIN (unique value Amazon uses to identify each item) of the book.
Rank	A number from 1 to 100 depending on the rank of the book upon the scan.
Review Score	A decimal between 0 and 5 which is the average score given by users to the book.
Number of Reviews	The number of reviews at the time of scanning.
Price	The price of the book in £.

Additionally when scanning the top 100 the date and time of scan initiation will be saved.

The webpages for specific books was derived from the partial URL

“<https://www.amazon.co.uk/dp/>” followed by the ASIN of the book. The data scraped from each specific webpage is as follows:

Data	Description
Title of the Book	The title of the book.
ASIN	The ASIN (unique value Amazon uses to identify each item) of the book.
Author of the Book	The name of the author of the book.
URL of the Book	The URL of the book on the kindle store. Used when outputting data to the user and when scraping the top 100.
Length of the Book	The number of pages in the book.
File Size of the Book	The file size of the book, normally in KB.
Language of the Book	The language that the book is written in.
Publisher of the Book	The publisher of the book.

An example of the format of the HTML to be scraped is as follows:

```

<div class="zg_itemimmersion" style="height:315px">
<div class="zg_rankDiv">
<span class="zg_rankNumber">
Ranking of book 21
</span>
</div>
<div class="zg_itemWrapper">
<div class="a-section a-spacing-none p13n-asin" data-p13n-asin-metadata="{\"ref\":\"zg_bs_digital-text_21\", \"asin\":\"B0769TDC8N\"}">
<a class="a-link-normal" href="\"THIRD-VICTIM-gripping-thriller-twists-ebook/dp/B0769TDC8N\"/ref=zg_bs_digital-text_21/262-1512731-4035123?_encoding=UTF8&psc=1&refRID=FKCBYNANA46XHRHE881P\"><div class="a-section a-spacing-mini"></div>
<div aria-hidden="true" class="p13n-sc-truncate-sky p13n-sc-truncated-hyphen p13n-sc-line-clamp-1" data-rows="1" data-truncate-mix-weblab="true">
HIS THIRD VICTIM a gripping crime thriller full of twists Title
</div>
</a>
<div class="a-row a-size-small"><span class="a-size-small a-color-base">HELEN H. DURRANT</span></div>
<div class="a-icon-row a-spacing-none">
<a class="a-link-normal" href="/product-reviews/B0769TDC8N/ref=zg_bs_digital-text_cr_21/262-1512731-4035123?ie=UTF8&refRID=FKCBYNANA46XHRHE881P" title="4.6 out of 5 stars">
4.6 out of 5 stars
</a>
</div>
<div class="a-row a-size-small"><span class="a-size-small a-color-base">73</span></div>
<div class="a-row a-size-small"><span class="a-size-small a-color-secondary">Kindle Edition</span></div><div class="a-row"><span class="a-size-base a-color-price"><span class="p13n-sc-price">£1.99</span></span></div>
</div>
</div>

```

Examples of data to be extracted have been highlighted.

#### 4.1: Ethical Use of Data

All data scraped by the program is in the public domain, and therefore Research ethics approval was not required for this project. The data can be accessed by anyone with an internet connection and a computer.

When referring to a human subject:

“1. A *human subject* is defined as a living individual about whom an investigator conducting research obtains (1) data through intervention or interaction with the individual, or (2) identifiable private information about whom includes a subject’s opinion on a given topic.”

This definition was obtained from RDIA [11].

This project does not obtain data from intervention or interaction with specific individuals and does not store any identifiable private information regarding opinions on any given topic, aside from feedback given by the project supervisor and marker which was given freely with their knowledge and consent.

## **5.0: Design**

### **5.1: Summary of Proposal**

The background, aims and objectives of this project have been outlined largely in the Specification document, but are repeated below.

The intention of the project is to create a program which will track the rank of items in the Amazon Kindle top 100 list over time using web scraping. The project is being completed for the project supervisor:[].

The proposed solution is a program, written in Python, which will scrape the kindle top 100 list for details on the books in the top 100 at a given time, and will also, by use of subsequent scans, keep track of the position of books as they go through the top 100 list over time. The system will additionally allow the user to search for a specific book and will also store relevant data, such as time in the top 100 and basic information about the book, in a database.

The system will:

- Allow the user to scrape the Amazon Kindle top 100 and extract details
- Allow the user to scrape the Amazon Kindle top 100 to find a specific book, given its URL
- Store basic information, URL and current rank of each item in a database
- Show the user an overview of a specific item such as; how long it has been in the top 100; how long it has been at its current rank; the item's highest historical rank; and dates/times of when data was retrieved.

Since the creation of the initial design document, however, a number factors have resulted in the design being changed. Minor changes and improvements as a result of testing and prototyping can be found later in this document and in the appendices for each prototype produced. The most major of changes to the final design compared to the initial design can be found below:

- It was decided during the creation of the program that it would be interesting to store any changes of price a book may have so further analysis could be performed, such as seeing if there are spikes in the number of reviews when a price decreases, or to see how (or if) the average review score of a book changes depending on the price. Previously the system stored price as an unchanging value along with the book. Changes made to the system resulted in the price of the books being stored in the results table instead of the books table to allow the storage of multiple values for prices.
- During the creation of the GUI, a number of popular Python modules were attempted to be used, namely PyQt and Tkinter. Due to timing constraints and unfamiliarity with the language as a whole it was decided that, instead of using new tools to create a GUI, a GUI would be created in Visual Basic and call the Python code from separate files. The main code for the project was therefore split into separate

files rather than the entire system being in a single file as previously intended. This has made the system perhaps less portable and more difficult to assemble, but has resulted in it being much easier to locate what part of the program is running at any given time.

- The option to omit certain books from being included in scans has been removed entirely. It was decided that this was not necessary as the user could potentially be presented with a list of 100 unfound books after a scan and must manually decide which ones they no longer wish to search for. Instead the system now scrapes as much data as possible at all times. The user is also not informed that a book was not found in the top 100 for the similar reasons; being presented with a list of possibly 100 now irrelevant books is not user friendly.
- The synopsis of each book was intended to be scraped, yet attempting to scrape this information proved to be unsuccessful. Instead additional columns have been added to the Books table to store additional information from a specific book's page. These are; the file size of the book; the language of the book; and the publisher of the book. These values were found on every book page checked so will likely always get a value when searching for this information. The program can also be relatively easily altered to get data that is not guaranteed to be on every page, but this was not implemented. Details on this implementation can be found in the realisation and evaluation section of this document.
- The design of the classes for use in the program has changed. Inheritance was intended to be used but this was replaced by separate classes to reduce code fragmentation among the split Python files.

## **5.2: Description of System**

For normal scanning, the program will first scrape the top 100 list and extract each item. If a book has been found during scanning for the first time, the unchanging details of the books will be added to the Books table. A new result in the Results table is then created for each book found in the scan.

Upon the next scan, if the items are found in the top 100 again then a new record is created in the Results table with updated values.

Alternatively, the user can also add a specific book. This would be done by allowing the user to input a specific URL or ASIN (this being the unique ID that Amazon uses to identify items) of an item. The correct URL is checked and basic information about the item is added to the tables as described above, assuming it is not already in the database.

Specific books can also be searched for. The user would select a specific book from a list showing the contents of the database and the program will then scrape the top 100 list only looking for that specific book.

The system will be comprised of two different sections:

1. The User Program- this is how the user will interact with the system. Written in Python and accompanied by a UI written in Visual Basic.

2. Database- this is where the information that is scraped is stored. This will be a MySQL database which will be queried indirectly by the user through the use of the program.

### **5.3: Data Dictionaries**

Any length of the columns that have changed from the initial design documentation has been done so after testing determined the lengths to be too small.

Books(**BOOKID**, ASIN, BookTitle, BookAuthor, BookURL, BookLength, BookFileSize, BookLanguage, BookPublisher)

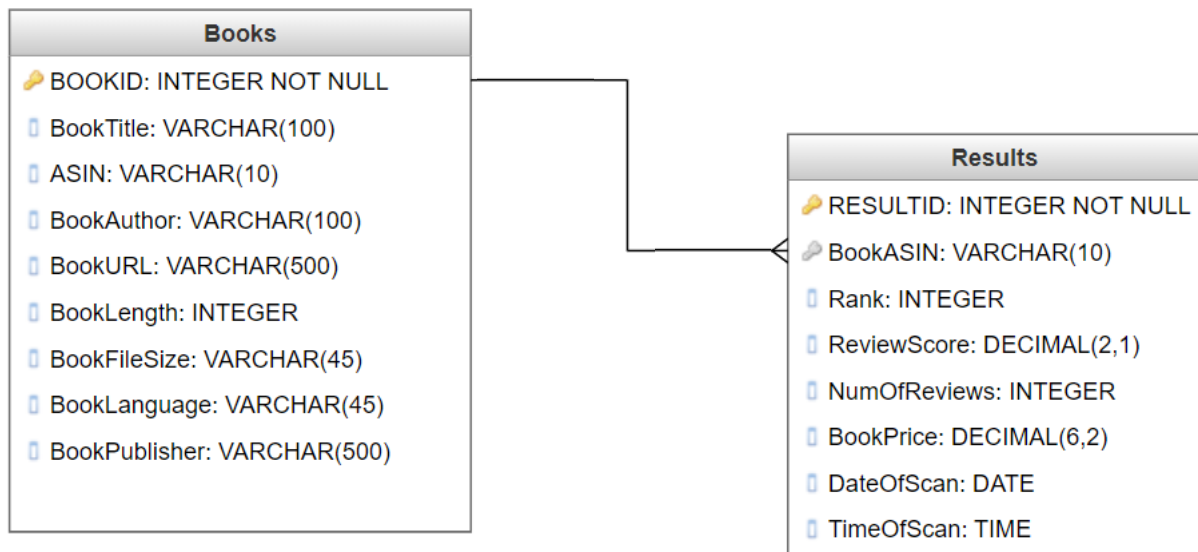
Name	Type	Length	Functionality
BOOKID	INT NOT NULL	NOT NULL	Primary key for Books table. Used to uniquely identify each record in this table and is also used as a foreign key in the Results table. Autoincrements.
BookTitle	VARCHAR	100	The title of the book. Used when outputting data to the user.
ASIN	VARCHAR	10	The ASIN (unique value Amazon uses to identify each item) of the book.
BookAuthor	VARCHAR	200	The name of the author of the book.
BookURL	VARCHAR	100	The URL of the book on the kindle store. Used when outputting data to the user and when scraping the top 100.
BookLength	INT	4	The number of pages in the book.
BookFileSize	VARCHAR	45	The file size of the book, normally in KB.
BookLanguage	VARCHAR	45	The language that the book is written in.
BookPublisher	VARCHAR	500	The publisher of the book.

*A record can't be deleted from Books without first having any corresponding records deleted in the Results table. This will be accomplished through cascading deletes.*

Results(**ResultID**, **BookASIN**, Rank, ReviewScore, NumOfReviews, BookPrice, DateOfScan, TimeOfScan)

Name	Type	Length	Functionality
ResultID	INT NOT NULL	NOT NULL	Primary key for Results table.
BookASIN	VARCHAR	NOT NULL	Foreign key of "ASIN" from the Books table. Used to separate the details of the book from the results table in order to reduce data redundancy.
Rank	INT	2	A number from 1 to 100 which is set depending on the rank of the book upon the scan.
ReviewScore	DECIMAL	(2,1)	A decimal between 0 and 5 which was the average score given by users to the book.
NumOfReviews	INT	4	The number of reviews at the time of scanning.
BookPrice	DECIMAL	(6,2)	The price of the book in £.
DateOfScan	DATE	8	The date of the scan (YYYY-MM-DD).
TimeOfScan	TIME	4	The time of the scan (HH:MM:SS).

## 5.4: Logical Table Design



## 5.5: Physical Table Design

BOOKID	1
BookTitle	An Honest Citizen
ASIN	B012RGSEP4
BookAuthor	Laura Byder
BookURL	<a href="https://www.amazon.co.uk/An-Honest-Citizen-ebook/dp/B012RGSEP4/">https://www.amazon.co.uk/An-Honest-Citizen-ebook/dp/B012RGSEP4/</a>
BookLength	567
BookFileSize	5000 KB
BookLanguage	English
BookPublisher	Petrel

RESULTID	1
BookASIN	B012RGSEP4
Rank	1
ReviewScore	4.5
NumOfReviews	28
BookPrice	0.99
DateOfScan	09/10/2016
TimeOfScan	13:45:23

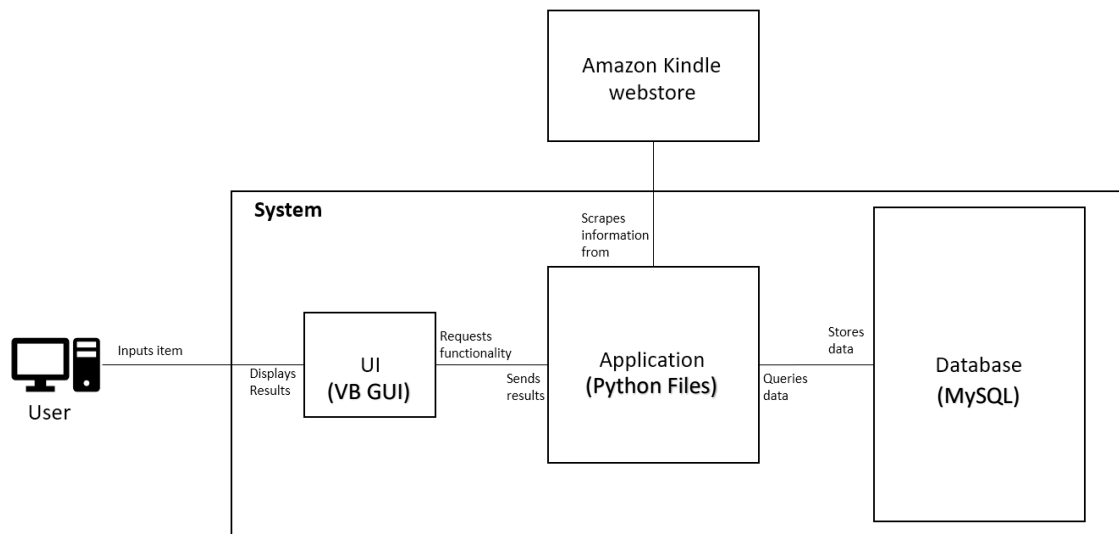
RESULTID	59
BookASIN	B012RGSEP4
Rank	3
ReviewScore	4.4
NumOfReviews	31
BookPrice	0.99
DateOfScan	19/10/2016
TimeOfScan	50

RESULTID	684
BookASIN	B012RGSEP4
Rank	47
ReviewScore	4.6
NumOfReviews	189
BookPrice	1.99
DateOfScan	01/11/2016
TimeOfScan	08:30:10

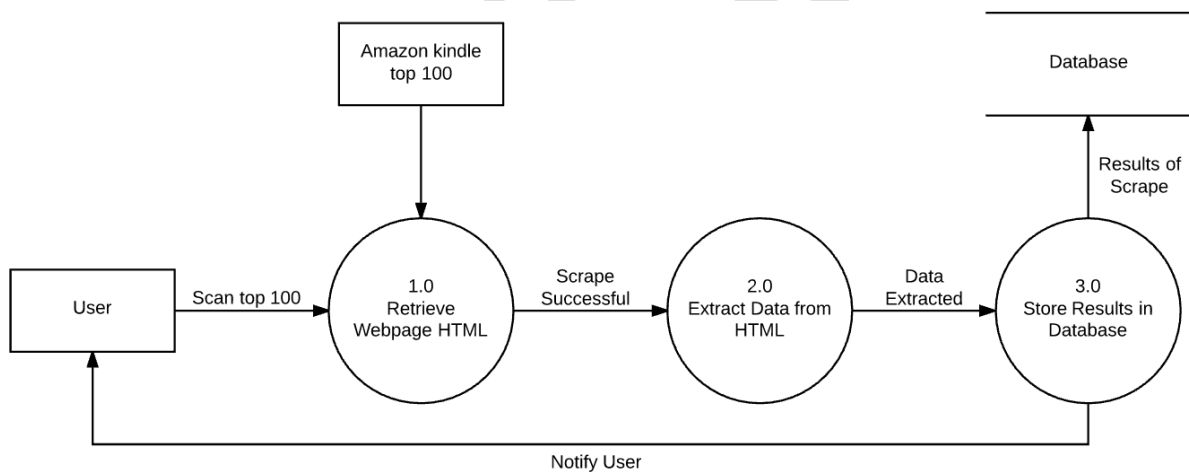
Note that this is synthetic data.

## 5.6: System Boundary Diagram

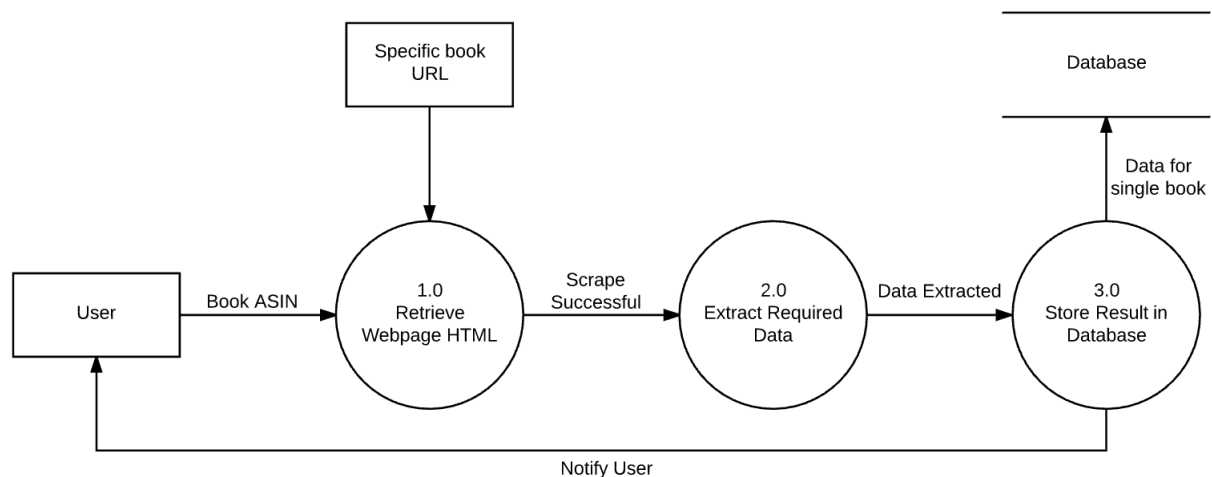


## 5.7: Data Flow Diagrams

### 5.7.1: Amazon kindle top 100 Scrape



### **5.7.2: Specific Book Search for Additional Information**



Note here that the book ASIN can either be input directly from the user for a new book or chosen from a list of currently existing books in the database.

### **5.8: Files Produced**

Multiple Python files will be produced. Each will perform a specific function and be called by the UI.

#### **5.8.1: Top 100 Scrape**

Code called when the user wants to perform a new scrape of the top 100. Accesses each web page in the Amazon kindle top 100, saves the HTML to a variable, extracts the required data from the page, saves to the database.

#### **Pseudocode**

Create list of 100 books

For  $i = 0$  to 4, scrape the data for the  $i^{\text{th}}$  page, starting at page 0:

- Get the current date/time and save to the  $(i \text{ to } i+20)$  books in the list
- Get the rankings of each book and save to the  $(i \text{ to } i+20)$  books in the list
- Get the titles of each book and save to the  $(i \text{ to } i+20)$  books in the list
- Get the ASINs of each book and save to the  $(i \text{ to } i+20)$  books in the list
- Get the author of each book and save to the  $(i \text{ to } i+20)$  books in the list
- Get the price of each book and save to the  $(i \text{ to } i+20)$  books in the list
- Get the review scores of each book and save to the  $(i \text{ to } i+20)$  books in the list
- Get the number of reviews of each book and save to the  $(i \text{ to } i+20)$  books in the list
- Get the URL for each book and save to the  $(i \text{ to } i+20)$  books in the list

For  $i = 0$  to 100

- Insert relevant information into the Books table
- Insert relevant information into the Results table

#### **Classes used**



resultsClass: used to store the result of an individual book in the top 100 scan

resultsClass
bookTitle: String
bookASIN: String
bookAuthor: String
bookURL: String
bookPrice: String
bookRank: Integer
bookReviewScore: String
bookNumOfReviews: Integer
bookDateOfScan: String
bookTimeOfScan: String

A list of 100 resultsClass is generated to store the results of the scan.

### MySQL usage

For inserting relevant information into Books table:

```
INSERT INTO books (BookTitle, ASIN, BookAuthor, BookURL) VALUES (%s,%s,%s,%s)
```

Where %s is set to the correct value from the list of results. If the book already exists, i.e the program throws exception with code 1062, the user is told and the book is not created.








For inserting relevant information into Results table:

```
INSERT IGNORE INTO results (BookASIN, Rank, ReviewScore, NumOfReviews, BookPrice, DateOfScan, TimeOfScan) VALUES (%s,%s,%s,%s,%s,%s,%s)
```

Where %s is set to the correct value from the list of results. This differs from the book insertion by using the IGNORE keyword as it is not possible in practice to have duplicates and therefore we do not need to check for them. Duplicates could only arise from starting two scans at the exact same time which the GUI does not allow for. This will need to be improved further in future versions to catch any other errors that may arise.

### 5.8.2: Specific Book Scrape

Used when scraping the Amazon kindle top 100 for a specific book. Checks whether the book is in the database already and if not adds it, checks each page of the top 100 for the book and if it is found creates a new result in the results table.

specificBookResultsClass
 bookASIN: String  bookPrice: String  bookRank: Integer  bookReviewScore: String  bookNumOfReviews: Integer  bookDateOfScan: String  bookTimeOfScan: String

### **Pseudocode (assuming input is ASIN validated from GUI):**

```

Get ASIN as input
Connect to database
Attempt to find ASIN in results table
If ASIN was not found
    Generate URL from ASIN
    Check if passed in ASIN leads to expected page
    If page was as expected, call addBook file with ASIN
    If page was not as expected, tell user and break
End if
found = false
For each page in the top 100
    Attempt to find the ASIN in the page
    If ASIN is on the page
        found = true
    End if
    If found == true
        Break the for loop
    End if
Next page
If found == true
    Create new specificBookResultClass
    Scrape relevant information from the correct page
    Insert new record into database
End if

```

### **MySQL usage**

See top 100 scrape for SQL regarding adding a new result.

To check if book exists in the books table:

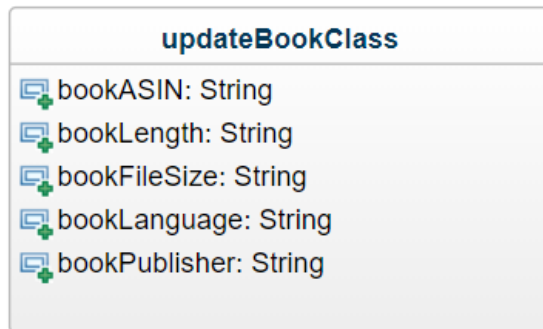
```
SELECT BOOKID FROM books WHERE ASIN = %s
```

Where %s is set to the passed in ASIN.

### 5.8.3 Update Book

Used when the user wants to get additional information about a book which can not be found on the Amazon kindle top 100 pages. ASIN is passed in (assumed correct from GUI), URL for the ASIN passed in is generated and accessed, HTML of the webpage is saved to a variable, relevant data is extracted, and the book is updated with new values in the database.

Relevant data is stored in a list in the HTML code so can be iterated through.



### Pseudocode

```
currentBook = new updateBookObject()
Generate URL from passed in ASIN
Access URL
Extract the list containing the relevant data in the HTML
For each element in the list
    If current element is length of the book
        currentBook.bookLength = current element
        continue
    If current element is file size of the book
        currentBook.bookFileSize = current element
        continue
    ...
    ...
    ...    Repeat for all elements in the list
    ...    for each required variable in currentBook
    ...
Next element
Update relevant book record with new information extracted
```

Only data which is guaranteed to be on every page will be extracted in this version of the program. The program can easily be updated to get more possible information on each book by inserting more if statements for potentially appearing data, but since this data is likely unchanging it is not too relevant to the system and could result in a large amount of irrelevant information.

### MySQL usage

```
""""UPDATE books SET BookLength=%s, BookFileSize=%s, BookLanguage=%s,
BookPublisher=%s WHERE ASIN=%s """"
```

Self explanatory.

#### **5.8.4 Delete Book**

Takes an ASIN as input and deletes the corresponding record from the books table. Due to cascading deletes, all associated records are deleted too. Does not require any additional classes or objects.

##### **Pseudocode**

Get ASIN as input  
Connect to database  
Delete from the books table the relevant record

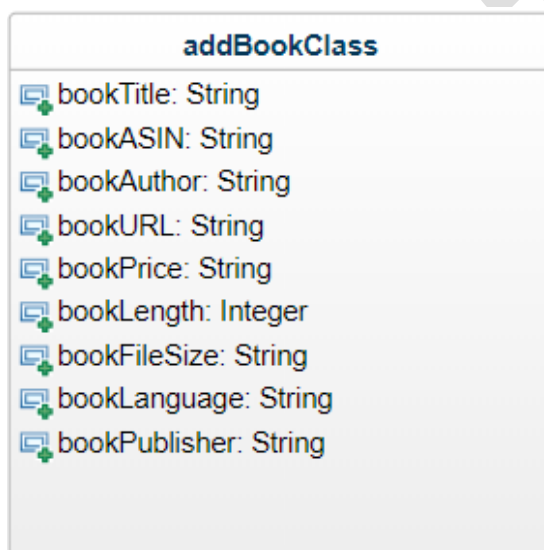
##### **MySQL usage**

```
""""DELETE FROM books WHERE ASIN = %s""""
```

Self explanatory.

#### **5.8.5 Add Book**

Takes ASIN as input, accesses relevant web page, extracts data and creates a new book with the scraped data.



##### **Pseudocode**

Get ASIN as input  
Generate URL  
Access URL  
`currentBook = new addBookClass()`  
set `currentBook.bookASIN` and `currentBook.bookURL`  
`currentBook.bookAuthor` = extracted author from HTML  
Extract rest of information in same manner as the loop in update book  
Attempt to add book to database

If book already exists  
    Update existing record with newly scraped data  
End if

### **MySQL usage**

When attempting to add a new book:

```
INSERT INTO books (BookTitle, ASIN, BookAuthor, BookURL, BookLength, BookFileSize,  
BookLanguage, BookPublisher) VALUES (%s,%s,%s,%s,%s,%s,%s,%s)
```

Where each %s is set to the relevant value from the currentBook object (an instance of addBookClass).

When updating an already existing record with new values:

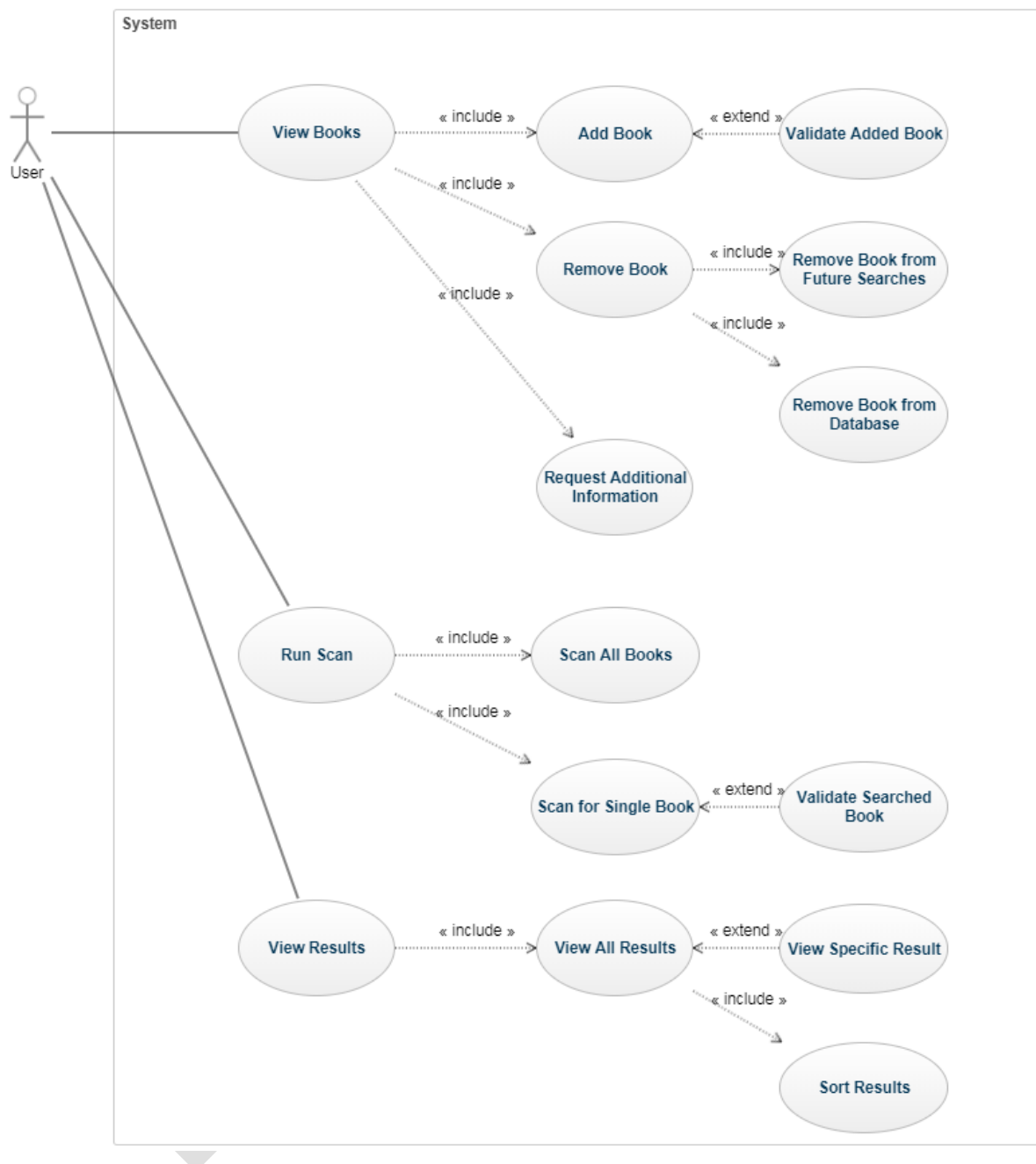
```
UPDATE books SET BookLength=%s, BookFileSize=%s, BookLanguage=%s, BookPublisher=%s  
WHERE ASIN=%s
```

Where each %s is set to the relevant value from the currentBook object (an instance of addBookClass).

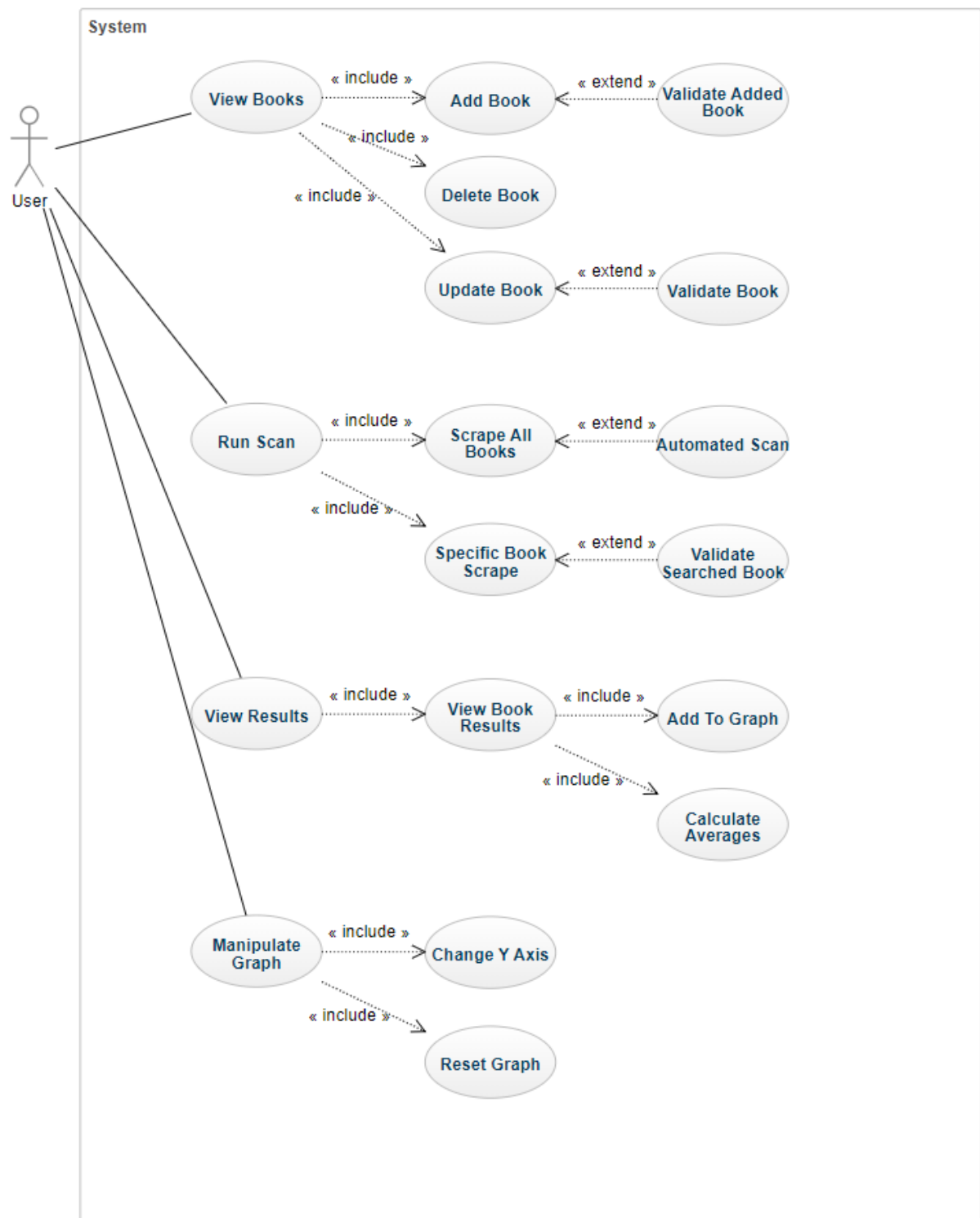
DONOT

## 5.9: Use Case Diagrams

### 5.9.1: Old Use Case Diagram



### 5.9.2: New Use Case Diagram



Changes from initial use case diagram:

- Remove Book consolidated into Delete Book
- Request Additional Information changed to Update Book
- View All Results changed to View Book Results
- View Specific Result and Sort Results removed

- Manipulate Graph, Change Y Axis and Reset Graph added.

#### 5.10: Use Case Tables

<b>ID</b>	<b>UC1</b>
<b>Name</b>	View Books
<b>Description</b>	User accesses the view books section of the program
<b>Pre-conditions</b>	None
<b>Event flow</b>	User is shown a view of the results and books in the database.
<b>Extension Points</b>	
<b>Triggers</b>	User clicks the Fill List Boxes button
<b>Post-condition</b>	User can proceed to perform actions on the results of scans and the books themselves

<b>ID</b>	<b>UC2</b>
<b>Name</b>	Add Book
<b>Description</b>	User attempts to add a new book the books database
<b>Pre-conditions</b>	User is on the books section of the program
<b>Event flow</b>	Include UC1 "View Books" User inputs the ASIN or URL of the book to be added to the database.
<b>Extension Points</b>	UC3 "Validate Added Book"
<b>Triggers</b>	User clicks the add book button
<b>Post-condition</b>	User has added a new book to the database

<b>ID</b>	<b>UC3</b>
<b>Name</b>	Validate Added Book
<b>Description</b>	The book the user is attempting to add is verified
<b>Pre-conditions</b>	User has inserted an ASIN or URL
<b>Event flow</b>	URL or ASIN is checked to see if it is valid. URL or ASIN is checked to see if it already exists in the database.
<b>Extension Points</b>	
<b>Triggers</b>	User has entered a URL or ASIN
<b>Post-condition</b>	User is told whether the book was added successfully or if the book was not added due to some failure

<b>ID</b>	<b>UC4</b>
<b>Name</b>	Delete Book
<b>Description</b>	The user wants to remove a book from the system
<b>Pre-conditions</b>	User is on the books section of the program
<b>Event flow</b>	User selects a book from a list.
<b>Extension Points</b>	
<b>Triggers</b>	User clicks the delete book button



<b>Post-condition</b>	Book is deleted from the database
-----------------------	-----------------------------------

<b>ID</b>	<b>UC5</b>
<b>Name</b>	Update Book
<b>Description</b>	A book in the database is updated with new values
<b>Pre-conditions</b>	User has selected a book
<b>Event flow</b>	Selected book's URL is accessed and extra data is scraped New data is saved to the book's record in the database
<b>Extension Points</b>	
<b>Triggers</b>	User clicks the update book button
<b>Post-condition</b>	Book is no longer searched for in future scans

<b>ID</b>	<b>UC6</b>
<b>Name</b>	Run Scan
<b>Description</b>	User decides what type of scan they want to perform
<b>Pre-conditions</b>	None
<b>Event flow</b>	User is taken to the scanning section of the program
<b>Extension Points</b>	
<b>Triggers</b>	
<b>Post-condition</b>	User can choose a type of scan

<b>ID</b>	<b>UC7</b>
<b>Name</b>	Scan All Books
<b>Description</b>	A full scan of the Amazon kindle top 100 is performed
<b>Pre-conditions</b>	User selects the scan all books option
<b>Event flow</b>	Include UC8 "Run Scan" A full scan of the Amazon kindle top 100 is performed New results and books are added to the database
<b>Extension Points</b>	
<b>Triggers</b>	User selects the option to do a full scan
<b>Post-condition</b>	A full scan has been performed

<b>ID</b>	<b>UC8</b>
<b>Name</b>	Automated Scan
<b>Description</b>	A full scan of the Amazon kindle top 100 is performed at a given time
<b>Pre-conditions</b>	None
<b>Event flow</b>	Include UC7 "Scan All Books" A full scan of the Amazon kindle top 100 is performed New results and books are added to the database
<b>Extension Points</b>	
<b>Triggers</b>	A time is reached when a new scan will be performed
<b>Post-condition</b>	A full scan has been performed

<b>ID</b>	<b>UC9</b>
<b>Name</b>	Specific Book Scrape
<b>Description</b>	A scan of the Amazon kindle top 100 is performed to find a single book
<b>Pre-conditions</b>	User selects the scan for specific book
<b>Event flow</b>	Include UC7 "Scan All Books" User selects a book to be scanned for A scan of the Amazon kindle top 100 is performed Any results related to the subject of the search are stored in the database
<b>Extension Points</b>	UC11 "Validate Searched Book"
<b>Triggers</b>	User selects the option to scan for a single book
<b>Post-condition</b>	Any (if any) new records related to the chosen book are added to the database

<b>ID</b>	<b>UC10</b>
<b>Name</b>	Validate Searched Book
<b>Description</b>	The selected book is validated
<b>Pre-conditions</b>	User selects the scan for specific book User has selected a book
<b>Event flow</b>	The selected book is checked to see if it is in the results If it is not, then the scan is terminated, and the user is told so
<b>Extension Points</b>	
<b>Triggers</b>	User has selected a book to be searched for
<b>Post-condition</b>	Scan is terminated, and no new records are created

<b>ID</b>	<b>UC11</b>
<b>Name</b>	View Results
<b>Description</b>	User views the results of searches
<b>Pre-conditions</b>	
<b>Event flow</b>	User is shown a view of the results in the database
<b>Extension Points</b>	
<b>Triggers</b>	
<b>Post-condition</b>	User can view results

<b>ID</b>	<b>UC12</b>
<b>Name</b>	View Book Results
<b>Description</b>	User views the results for a specific book
<b>Pre-conditions</b>	User is on the results section of the program User is viewing all results
<b>Event flow</b>	User chooses the ASIN of a book from a list User is shown only the results of that book
<b>Extension Points</b>	
<b>Triggers</b>	User selects the view specific book results option

<b>Post-condition</b>	User is viewing results for a specific book
-----------------------	---

<b>ID</b>	<b>UC13</b>
<b>Name</b>	Add To Graph
<b>Description</b>	User adds the results of the currently selected book to a graph
<b>Pre-conditions</b>	User is on the results section of the program User is viewing results for a book
<b>Event flow</b>	Include UC12 "View All Results" User selects an option to sort by Results are sorted depending on that option
<b>Extension Points</b>	
<b>Triggers</b>	User selects the option to sort the results
<b>Post-condition</b>	User is viewing a sorted list of results

<b>ID</b>	<b>UC16</b>
<b>Name</b>	View Single Result
<b>Description</b>	User views a single result
<b>Pre-conditions</b>	User is on the results section of the program User is viewing all results
<b>Event flow</b>	Include UC13 "View Book Results" User is shown that specific book's results on a graph
<b>Extension Points</b>	
<b>Triggers</b>	User clicks the add to graph button
<b>Post-condition</b>	Graph is updated with new values

<b>ID</b>	<b>UC17</b>
<b>Name</b>	Calculate Averages
<b>Description</b>	User sees additional information on the results of a book
<b>Pre-conditions</b>	User is on the results section of the program User is viewing the results of a specific book
<b>Event flow</b>	Include UC13 "View Book Results" User is shown that specific book's results' averages
<b>Extension Points</b>	
<b>Triggers</b>	User clicks the calculate averages button
<b>Post-condition</b>	List Box containing averages now contains new values

<b>ID</b>	<b>UC18</b>
<b>Name</b>	Manipulate Graph
<b>Description</b>	User manipulates the graph on the UI
<b>Pre-conditions</b>	User has selected a book
<b>Event flow</b>	User is given options to change the graph
<b>Extension Points</b>	
<b>Triggers</b>	None
<b>Post-condition</b>	User can edit the graph

<b>ID</b>	<b>UC19</b>
<b>Name</b>	Change Y Axis
<b>Description</b>	User changes the value on the Y axis of the graph
<b>Pre-conditions</b>	User is editing the graph
<b>Event flow</b>	User selects the new Y axis value from a list Graph is updated to show what the user selects
<b>Extension Points</b>	
<b>Triggers</b>	Change Y Axis button is clicked
<b>Post-condition</b>	Graph now shows a different Y axis

<b>ID</b>	<b>UC20</b>
<b>Name</b>	Reset Graph
<b>Description</b>	User removes all data currently on the graph
<b>Pre-conditions</b>	There are results plotted on the graph
<b>Event flow</b>	All results currently plotted on the graph are removed
<b>Extension Points</b>	
<b>Triggers</b>	Reset Graph button is clicked
<b>Post-condition</b>	Graph is empty

### **5.11: Evaluation Design**

The final system will be evaluated against the following criteria:

- Does the system successfully scrape the Amazon kindle top 100 list?
- Does the system successfully find details of a book given the correct URL?
- Does the system store all the necessary information for each book?
- Does the system provide sufficient analysis on any given book?
  - Does the system show how long a book has been in the top 100?
  - Does the system show how long a book has been at its current rank?
  - Does the system show the book's highest historical rank (since the first time it was found in a scan)?
  - Does the system provide dates/times for specific scans?
  - Does the system provide functionality to see results of a specific scan on a book?
- Does the system have an adequate UI?

The system will be constantly evaluated through the development by use of prototypes. Each of these prototypes will require different evaluation criteria.

1. The first prototype will be evaluated by its creator and by the project supervisor from a technical standpoint. The system will be checked to see if the amount data it scrapes is sufficient to continue with the next stages of implementation, and that the data scraped is stored correctly and can be presented to the user.
2. The second prototype will be evaluated by its creator and by the project supervisor from a technical and client standpoint respectively, meaning the code itself will be

checked for correctness by myself along with the results of computation and analysis on the scraped data, and the project supervisor will decide if the specification has been met and provide feedback through questionnaires.

3. Any further prototypes will be tested by myself and additionally by the project supervisor in a similar fashion to the second prototype.

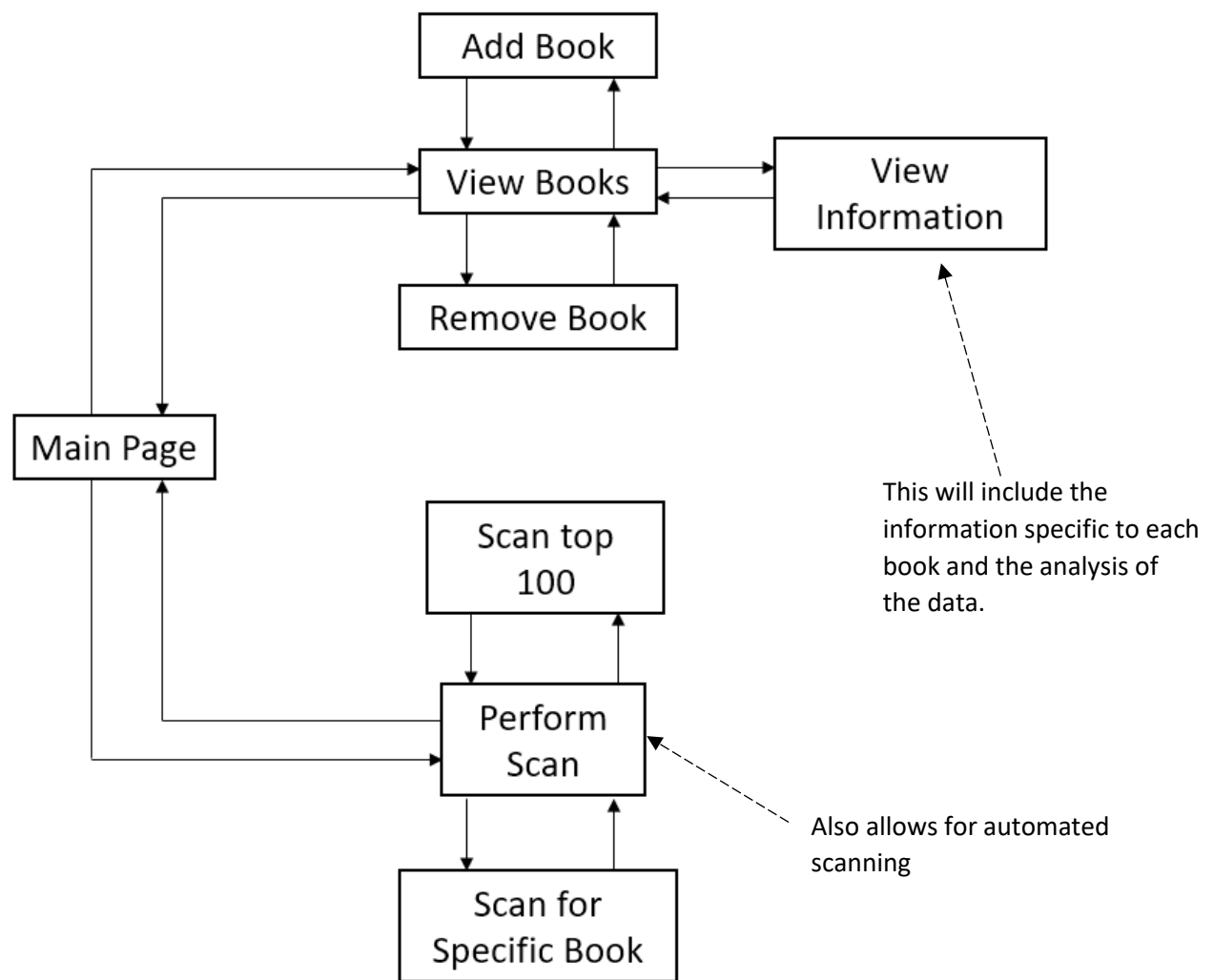
As for the more technical evaluation, each section of the program will be thoroughly tested using a number of test cases. These again will be developed further to the end of the project's development as to more closely tailor them to what is required at the time.

Test case example, for searching for a book:

Option	Value Input	Result	Expected result?
BookTitle	Nothing	No input detected	✓
BookTitle	"A Man and his Boat" (database does not contain such a book)	Book not found.	✓
BookTitle	"A Man and his Boat" (database does contain such a book)	The book found is correct, and user is presented with further options related to the book found.	✓

Similar testing will be carried out extensively with each section of the program at each prototype phase to fix any errors. The results of these tests along with screenshots of how each test was performed are included in the Realisation section of the final project report and in the testing document in the appendix.

### 5.12: Process Map



5.13: UI Design

Form1

Books:  
ListBox1

Results:  
ListBox2

Book Name: bookName  
BookID: ...  
ASIN: ...  
Author: ...  
URL: ...  
Length: ...  
File Size: ...  
Language: ...  
Publisher: ...

ResultID: ...  
ASIN: ...  
Rank: ...  
Review Score: ...  
# of Reviews: ...  
Price: ...  
Date of Scan: ...  
Time of Scan: ...

Legend1 - Empty

Fill List Boxes

Add Book

Update Book

Delete Book

New Scan

Specific Book Scan

ListBoxAverages

Calculate Averages

Show Current Book Results on Graph

Change Y axis

Reset Graph

ListBox1 will either contain the ASINs of each book, in an attempt to reduce the amount of memory the program will need, or the titles of the books. If the title of the book is used then the GUI will need to store the titles along with the ASINs of the book in an object as only the ASIN is guaranteed to be unique; two books may share the same name, but two books will never have the same ASIN. If the ASIN is used, then it will be much more difficult for the user to select the exact book that they require.

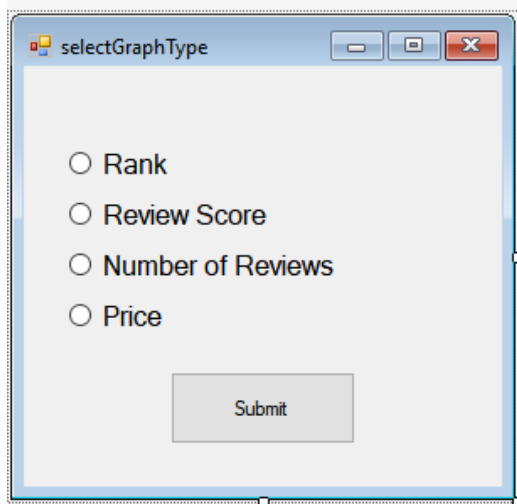
This trade-off will only be relevant when there is a very large number of books in the database. At this stage in the program development either of the options are suitable, but in an attempt to greatly improve usability in future versions the book names should be used.

ListBox2 will contain the ResultID of each result that the book has in the database.

The calculate averages button will calculate some addition information on the currently selected book and display the results in ListBoxAverages.

The large white box on the furthest right part of the UI is a graph tool to display all of the results of selected books, for comparison or for trend spotting.

The Change Y Axis button will allow the user to select what value they wish to be plotted on the Y axis of the graph, by the use of a pop-up menu:



### Important code

Most code is simply retrieving data from the database and manipulating the form objects included in Visual Studio. As such only code relevant to this particular system will be included in the design documentation, this being the SQL and a basic idea of which Python scripts are called from each part of the interface.

Each button will call the relevant .py file:

Button	.py Script Called
"Add Book"	addBook.py
"Update Book"	updateBook.py
"Delete Book"	deleteBook.py
"New Scan"	top100Scrape.py <b>OR</b> specificBookScrape.py

Note that these file names may differ in the final system.



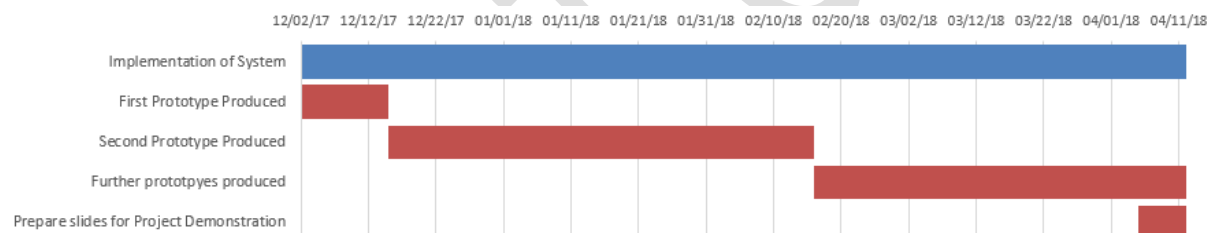
Any buttons not in the table above do not call any .py files.

### MySQL needed

Section of Interface	Trigger	Additional MySQL
Filling ListBox1	"Fill List Boxes" button is clicked	"SELECT * FROM books" – used to get the books from the database to show them in ListBox1
Updating the book details labels	New book is selected from ListBox1	"SELECT * FROM books WHERE ASIN = @ASIN" where @ASIN is replaced by the currently selected book's ASIN – used to get the data needed to update the labels
Adding a book to the graph	"Show Current Book Results on Graph" button is clicked	"SELECT * FROM results WHERE ResultID = @ResultID" – used to get the relevant results details in order to plot them on the graph

Any additional SQL used will be slight variants or the same (for example, filling ListBox2 with results will use the same SQL as adding a book to the graph) as the above and will not be included in this documentation.

### 5.14: Updated Gantt Chart for Implementation Phase



## **6.0: Realisation**

### **6.1: Prototype 1**

The aim of this system was essentially a feasibility check- to make sure that each page of the Amazon Kindle top 100 could indeed be scraped with the decided tools. Additionally, this prototype was developed to increase familiarity with the Python programming language and the HTML of the webpages. No database system was developed for this prototype.

Prototype 1 was started shortly after the design presentation. Python, the PyCharm IDE and the BeautifulSoup module were installed and used to create this version of the program. The prototype was finished on time and shown to the project supervisor.

The first step of creating the program was to see if the data on the webpage could be found in the HTML of the webpage, and therefore confirm it was not placed on the page dynamically, for example by a Javascript script executing after the webpage has loaded. Once the correct URL had been found the format of the URL was checked to see if any pattern was followed for each page to be scraped. It was found that the URLs for each page were identical apart from numbers indicating the current page number.

The URLs were then opened in the Google Chrome web browser and inspected manually. The HTML was searched for the data that would be scraped from each page and the locations noted for later use.

Next the main code for extracting the data was written. The data was found to be in a predictable order in the HTML, with the exception of the author of each book's name. The same class name was being used for each author's name and the phrase "Kindle Edition" so, instead of having to go through 20 instances of the HTML containing the required data, 40 instances would have to be iterated through. It was discovered that each even iteration of the loop (starting at 0) the author's name would be found and each odd iteration the irrelevant data would be found, and the pertinent code was adjusted.

The last discovery during this phase of development was that the URL for each book on Amazon's website can be derived from the string "https://www.amazon.co.uk/dp/" followed by the ASIN of the book. This made it considerably easier to obtain the URLs of each specific book: each URL could be generated after extracting the ASIN, so the URL did not have to be scraped from the page at all as long as the ASIN was scraped first.

Since this prototype was produced in an effort to better understand Python and the BeautifulSoup module, most errors encountered here were purely syntactical and not of enough interest to be included in this report.

This version of the program only scrapes one page of the Kindle top 100 and does not store the results of scans between program runs. To scrape more than one page the program must take a different hard coded value for the URL and be run separately for each page.

Example for code to scrape a certain piece of data:

```
# title first
currentItems = soup.find_all('div', {'class': 'p13n-sc-truncate p13n-sc-line-clamp-1'})
x = 0
for i in currentItems:
    # we only actually want 0 to 19
    if x < 20:
        i = i.string
        i = i.strip()
        resultList[x].bookTitle = i
        x = x + 1
    else:
        break
```

Data is stored in a "div" tag with class name specified

20 instances of the data are scraped in a loop

ResultsList is a list of length 20 of class resultsClass:

```
class resultsClass:
    # class to store the results of scans
    # note does not include book length or synopsis
    bookTitle = ""
    bookASIN = ""
    bookAuthor = ""
    bookURL = ""
    bookPrice = ""
    bookRank = ""
    bookReviewScore = ""
    bookNumOfReviews = ""
    bookDateOfScan = ""
    bookTimeOfScan = ""
```

For a simple run of the program on page 1:

```
resultsList = (list) <class 'list': [<__main__.resultsClass object at 0x00000163D083...
> 00 = (resultsClass) <__main__.resultsClass object at 0x00000163D0838DA0>
> 01 = (resultsClass) <__main__.resultsClass object at 0x00000163D0838E48>
> 02 = (resultsClass) <__main__.resultsClass object at 0x00000163D0838F00>
> 03 = (resultsClass) <__main__.resultsClass object at 0x00000163D0838F48>
> 04 = (resultsClass) <__main__.resultsClass object at 0x00000163D0838F88>
> 05 = (resultsClass) <__main__.resultsClass object at 0x00000163D0838FC8>
> 06 = (resultsClass) <__main__.resultsClass object at 0x00000163D0839000>
> 07 = (resultsClass) <__main__.resultsClass object at 0x00000163D0839048>
> 08 = (resultsClass) <__main__.resultsClass object at 0x00000163D0839088>
> 09 = (resultsClass) <__main__.resultsClass object at 0x00000163D08390C8>
> 10 = (resultsClass) <__main__.resultsClass object at 0x00000163D0839100>
> 11 = (resultsClass) <__main__.resultsClass object at 0x00000163D0839148>
> 12 = (resultsClass) <__main__.resultsClass object at 0x00000163D0839188>
> 13 = (resultsClass) <__main__.resultsClass object at 0x00000163D08391C8>
> 14 = (resultsClass) <__main__.resultsClass object at 0x00000163D0839200>
> 15 = (resultsClass) <__main__.resultsClass object at 0x00000163D0839248>
> 16 = (resultsClass) <__main__.resultsClass object at 0x00000163D0839288>
> 17 = (resultsClass) <__main__.resultsClass object at 0x00000163D08392C8>
> 18 = (resultsClass) <__main__.resultsClass object at 0x00000163D0839300>
> 19 = (resultsClass) <__main__.resultsClass object at 0x00000163D0839348>
__len__ = (int) 20
```

Each item in resultsList was as follows:

```

00 = {resultsClass} <__main__.resultsClass object at 0x00000163D0838DA0>
bookASIN = {str} 'B074P1B728'
bookAuthor = {str} 'Andrew Hart'
bookDateOfScan = {str} '2018-05-02'
bookNumOfReviews = {str} '3'
bookPrice = {str} '£3.99'
bookRank = {int} 1
bookReviewScore = {str} '3.5 out of 5 stars'
bookTimeOfScan = {str} '12-05'
bookTitle = {str} 'Lies That Bind Us'
bookURL = {str} 'https://www.amazon.co.uk/dp/B074P1B728'

```

with the each of the items in resultsList corresponding to the expected book.

This process was repeated for each page in the top 100 to ensure all could be scraped correctly. Additional screenshots will not be provided due to the repetitive nature of the results.

Page	Page URL	Result	Expected result?
1	https://www.amazon.co.uk/Best-Sellers-Kindle-Store/zgbs/digital-text/ref=zg_bs_pg_1?_encoding=UTF8&pg=1	Successful scrape	✓
2	https://www.amazon.co.uk/Best-Sellers-Kindle-Store/zgbs/digital-text/ref=zg_bs_pg_2?_encoding=UTF8&pg=2	Successful scrape	✓
3	https://www.amazon.co.uk/Best-Sellers-Kindle-Store/zgbs/digital-text/ref=zg_bs_pg_3?_encoding=UTF8&pg=3	Successful scrape	✓
4	https://www.amazon.co.uk/Best-Sellers-Kindle-Store/zgbs/digital-text/ref=zg_bs_pg_4?_encoding=UTF8&pg=4	Successful scrape	✓
5	https://www.amazon.co.uk/Best-Sellers-Kindle-Store/zgbs/digital-text/ref=zg_bs_pg_5?_encoding=UTF8&pg=5	Successful scrape	✓

## 6.2: Prototype 2

This version of the program uses the code from Prototype 1 in order to scrape all 5 pages of the Amazon Kindle top 100 and stores the results in a newly implemented database system.

The main aim of developing this prototype was to become acquainted with MySQL Notifier and to better understand how Python could be used alongside it. A new Python package has been introduced to allow this: MySQLdb.

The main discovery during the creation of this prototype was that books in the top 100 list did not necessarily have any reviews and consequently no review score either. This resulted in incorrect values being given to results during the scrape- if a book had no reviews then the HTML normally containing the review scores did not exist, the program would not realise this and assign the next book's review score to the current book instead. A new loop

was created to first find which results had no review scores and to skip that iteration of the loop. This is further explained by comments in the actual code file itself, but a quick overview is given below:

```
currentItems = soup.findAll('div', {'class': 'zg_itemImmersion'})
listOfSkips = []
x = 0
for i in currentItems:
    test = i.find('div', {'class': 'a-icon-row a-spacing-none'})
    print("\n")
    if test == None:
        #print("not found at location ")
        listOfSkips.append(x)
    x +=1

currentItems = soup.findAll('span', {'class': 'a-icon-alt'})
loopCounter = 0
x = 0
while loopCounter < 20:
    #if we come to an iteration that we want to skip, we must remain on the same currentItems[x] but increment the loop counter
    if loopCounter not in listOfSkips:
        currentValue = currentItems[x]
        currentValue = currentValue.string
        currentValue = currentValue.strip()
        # similarly to the price, we need to do a bit of string manipulation
        # we have a result in the format '3.9 out of 5 stars' when we need a decimal '3.9' to insert into the database
        currentValue = float(currentValue[:3])
        resultsList[loopCounter].bookReviewScore = currentValue
        x+=1
    loopCounter += 1
```

Create a list of indexes to skip

Skip indexes during extraction

Once this error was fixed the main loop of the program was written, where doScrape is the same code as found in Prototype 1 with errors fixed:

```
finalResultsList = [resultsClass() for i in range(0,100)]
currentURL = 'https://www.amazon.co.uk/Best-Sellers-Kindle-Store/zgbs/digital-text/ref=zg_bs_pg_1?_encoding=UTF8&pg=1'
doScrape(currentURL, 0)
time.sleep(1)
currentURL = 'https://www.amazon.co.uk/Best-Sellers-Kindle-Store/zgbs/digital-text/ref=zg_bs_pg_2?_encoding=UTF8&pg=2'
doScrape(currentURL, 1)
time.sleep(1)
currentURL = 'https://www.amazon.co.uk/Best-Sellers-Kindle-Store/zgbs/digital-text/ref=zg_bs_pg_3?_encoding=UTF8&pg=3'
doScrape(currentURL, 2)
time.sleep(1)
currentURL = 'https://www.amazon.co.uk/Best-Sellers-Kindle-Store/zgbs/digital-text/ref=zg_bs_pg_4?_encoding=UTF8&pg=4'
doScrape(currentURL, 3)
time.sleep(1)
currentURL = 'https://www.amazon.co.uk/Best-Sellers-Kindle-Store/zgbs/digital-text/ref=zg_bs_pg_5?_encoding=UTF8&pg=5'
doScrape(currentURL, 4)
```

Note the second of waiting between each scrape in an attempt to avoid bot detection.

Once the main loop had been tested the database system was implemented. The design of this database can be found in the original design document, but has been included below:

Books table, to store the information on each book found in the scans:

Column	Type	Default Value	Nullable	Character Set	Collation	Privileges	Extra
BOOKID	int(11)		NO			select,insert,update,references	auto_increment
BookTitle	varchar(150)		YES	utf8	utf8_general_ci	select,insert,update,references	
ASIN	varchar(10)		YES	utf8	utf8_general_ci	select,insert,update,references	
BookAuthor	varchar(200)		YES	utf8	utf8_general_ci	select,insert,update,references	
BookURL	varchar(100)		YES	utf8	utf8_general_ci	select,insert,update,references	
BookPrice	decimal(6,2)		YES			select,insert,update,references	
BookLength	int(11)		YES			select,insert,update,references	
BookSynopsis	varchar(1500)		YES	utf8	utf8_general_ci	select,insert,update,references	
IncludedInScans	tinyint(4)	1	NO			select,insert,update,references	

Results table, to store the information gained from each scrape:

Column	Type	Default Value	Nullable	Character Set	Collation	Privileges	Extra
ResultID	int(11)		NO			select,insert,update,references	auto_increment
BookASIN	varchar(10)		NO	utf8	utf8_general_ci	select,insert,update,references	
Rank	int(11)		YES			select,insert,update,references	
ReviewScore	decimal(2,1)		YES			select,insert,update,references	
NumOfReviews	int(11)		YES			select,insert,update,references	
DateOfScan	date		YES			select,insert,update,references	
TimeOfScan	time		YES			select,insert,update,references	

Here BookASIN, from the results table, is a foreign key of ASIN from the books table. A full SQL dump of the database can be found in the appendix.

The database was initially set up on the University of Liverpool's own MySQL servers and Linux based operating systems, but it was decided that hosting the database locally would make the system easier to develop, due to being able to use MySQL Notifier on a Windows based operating system which I was more familiar with, and reduced reliance on systems outside of the scope of the project (these being the University's servers). After this switch the database was developed with no notable errors.

The next step in development was getting the Python code to interact with the database. Through the use of MySQLdb's official documentation [12] this step was completed, again with most errors being syntactical in nature.

One notable error which did occur was the system did not correctly store the prices of books. Instead of the correctly scraped values, the BookPrice column of the database contained "0.00" for all tuples:

BOOKID	BookTitle	ASIN	BookAuthor	BookURL	BookPrice	BookLength	BookSynopsis	IncludedInScans
1	The Note: The book everyone's talking about	B07175T6GY	Zoe Folbioq	https://www.amazon.co.uk/do/B07175T6GY	0.00	NULL	NULL	1
2	The Child: The must-read Richard and Judy Boo...	B01LBEDIHK	Fiona Barton	https://www.amazon.co.uk/do/B01LBEDIHK	0.00	NULL	NULL	1
3	The Tuscan Child	B074OL7WNM	Rhys Bowen	https://www.amazon.co.uk/do/B074OL7WNM	0.00	NULL	NULL	1
4	Then She Was Gone	B01MO1VL1Z	Lisa Jewell	https://www.amazon.co.uk/do/B01MO1VL1Z	0.00	NULL	NULL	1
5	Close to Home: The 'impossible to put down' Ric...	B071Z3Z1KN	Cara Hunter	https://www.amazon.co.uk/do/B071Z3Z1KN	0.00	NULL	NULL	1
6	The Lion, the Witch and the Wardrobe (The Chr...	B002UJ5J72	C. S. Lewis	https://www.amazon.co.uk/do/B002UJ5J72	0.00	NULL	NULL	1
7	Eleanor Oliphant is Completely Fine: Debut Sun...	B01MAYG70K	Gail Honevman	https://www.amazon.co.uk/do/B01MAYG70K	0.00	NULL	NULL	1
8	The Horse and His Boy (The Chronicles of Narni...	B002UJ5JHC	C. S. Lewis	https://www.amazon.co.uk/do/B002UJ5JHC	0.00	NULL	NULL	1
9	Prince Caspian (The Chronicles of Narnia. Book 4)	B002UJ5JM2	C. S. Lewis	https://www.amazon.co.uk/do/B002UJ5JM2	0.00	NULL	NULL	1
10	The Voyage of the Dawn Treader (The Chronid...	B00ALKNFJC	C. S. Lewis	https://www.amazon.co.uk/do/B00ALKNFJC	0.00	NULL	NULL	1
11	The Last Battle (The Chronicles of Narnia. Book 7)	B002UJ5JCW	C. S. Lewis	https://www.amazon.co.uk/do/B002UJ5JCW	0.00	NULL	NULL	1
12	The Silver Chair (The Chronicles of Narnia. Book 6)	B002UJ5JGI	C. S. Lewis	https://www.amazon.co.uk/do/B002UJ5JGI	0.00	NULL	NULL	1

On closer inspection the program was found to be throwing a warning each time a new record was added to the database: Warning: (1366, "Incorrect decimal value: '£2.89' for column 'BookPrice' at row 1"). This was fixed by removing the '£' sign from the value scraped and then converting it into a float.

```
#bookPrice is now in the format '£1.23'
#bookPrice is actually a decimal value, so we need to get rid of the pound sign
i = float(i[1:])
```

A similar problem with review scores not appearing correctly, solved with:

```
currentValue = currentValue.strip()  
# similarly to the price, we need to do a bit of string manipulation  
# we have a result in the format '3.9 out of 5 stars' when we need a decimal '3.9' to insert into the database  
currentValue = float(currentValue[:3])
```

Another error arose when attempting to insert a character into the database which was outside of the default encoding used by Python:

“UnicodeEncodeError: 'latin-1' codec can't encode character '\u2019' in position 12: ordinal not in range(256)”.

This was fixed by utilising the character set UTF8 instead.

```
db.set_character_set('utf8')  
c.execute('SET NAMES utf8;')  
c.execute('SET CHARACTER SET utf8;')  
c.execute('SET character_set_connection=utf8;')
```

The last noteworthy result of this prototype's development was that it was found to be not possible, using the current tools available to the program, to retrieve the synopsis of the book. Synopses were stored in an iframe in the HTML with a hidden source. No synopses were extracted during this version of the prototype and plans were made to extract some other data in the next version of the prototype.

This prototype was also not shown to the project supervisor, but formed the basis of the progress report, which was discussed in person to get feedback.

### **6.3: Prototype 3**

The main aim of this prototype was to introduce additional functionality to the scrape. New methods were introduced to:

- Add a book to the database based on an ASIN or URL (addBook)
- Update a book with new values from its specific URL (updateBook)
- Scan the top 100 for a specific book (specificBookTop100Scrape)
- Delete a book from the database (deleteBook)

Each of these methods are contained in a single file and instructions on how to execute each section of the program can be found at the bottom of the file.

Any additional objects for data storage during scans are slight variations of the already existing object definitions but with various different variable names for storage of different data scraped by each method.

The main top 100 scrape was slightly updated to allow for date and time logging of each scrape.

The price of the books was also moved from the books table to the results table as it was decided that noting book price changes could result in more trends being spotted, such as number of review increases with price decreases.



### 6.3.1: addBook

The first new feature introduced was the ability to add a book to the database. The HTML of a Kindle e-book on Amazon's website was inspected and the location of data to be extracted was found. It was found that the data could be extracted quite similarly to the code used to scrape the top 100 but with tags and class names changed, so this code will not be included here.

When attempting to locate the author's name in the HTML of the webpage, it was discovered that it could be in one of two places. To fix this a check was added when searching for the first location, and if it is not there then the other possible location it could be is checked.

A list was found in the HTML of the webpage which stored additional information on each book. This data would be extracted in a manner differently from previous scraping methods, as each item in the list was not guaranteed to be there and as such existence needed to be checked first.

```
currentItem = soup.find('div', {'class': 'content'})
currentItem = currentItem.find("ul")
for li in currentItem.find_all("li"):
    #can get a bunch of information here, number of pages is not guaranteed to be on there though so we must check to see if it is there
    #when looking for more information that could be added check this bit
    #seems that file size is always stored, as well as the language
    found = 0
    currentText = li.text
    if "Print Length:" in currentText:
        #we have found the print length string in the list
        currentBook.bookLength = currentText[14:17]
        continue
    if "File Size:" in currentText:
        #we have found the file size string in the list
        currentBook.bookFileSize = currentText[11:]
        continue
    if "Language:" in currentText:
        #we have found the language string
        currentBook.bookLanguage = currentText[10:]
        continue
    if "Publisher:" in currentText:
        #found the publisher string
        #this one is not on every page so may still be null after this loop
        currentBook.bookPublisher = currentText[11:]
        continue
```

Goes through the entirety of the list and extracts needed values. Attempts to extract data which, after a small amount of research, was found to be on every page (file size, language and print length) along with a piece of data not guaranteed to be on every page (print length and publisher). This code can be easily expanded to search for more items.

In order to store the newly scraped values in the database, the BookSynopsis column was dropped and new columns BookLength, BookFileSize, BookLanguage and BookPublisher were added. The IncludedInScans column was also removed due to it being unnecessary.

New books table:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
🔑 BOOKID	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
🔹 BookTitle	VARCHAR(500)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
🔹 ASIN	VARCHAR(10)	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
🔹 BookAuthor	VARCHAR(200)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
🔹 BookURL	VARCHAR(100)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
🔹 BookLength	INT(11)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
🔹 BookFileSize	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
🔹 BookLanguage	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
🔹 BookPublisher	VARCHAR(500)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

Test Runs:



Test	Input Data	Result	Expected result?
1	<a href="https://www.amazon.co.uk/dp/B07B7CFMBP/">https://www.amazon.co.uk/dp/B07B7CFMBP/</a> <ul style="list-style-type: none"> <li>Has file size and language</li> <li>Pass in a URL</li> </ul>	Book was successfully added to database	✓
2	<a href="#">B004T0A30A</a> <ul style="list-style-type: none"> <li>Has all information</li> <li>Passed in an ASIN</li> </ul>	Book was successfully added to database	✓

Screenshots of test runs can be found in testing section of the appendix.

### **6.3.2: deleteBook**

Next code to delete a book was created. This method consists solely of a simple delete query with a passed in ASIN and as such was developed without incident.

Test Runs:

Test	Input Data	Result	Expected result?
1	<a href="#">B004T0A30A</a> <ul style="list-style-type: none"> <li>Book in the database</li> </ul>	Book was deleted from database	✓
2	Nothing	Nothing happened	✓

### **6.3.3: specificBookTop100Scrape**

This new method takes an ASIN or URL as input and attempts to find the corresponding book in the top 100. If the book is not found in the database, it is added before continuing to scan the top 100.

Test	Input Data	Expected Result	Actual Result	Expected result?
1	"B074FYCSYQ"- a book <b>not</b> in the database but on the first page of the top 100.	A new record would be created in the books table and a new record would be created in the results table.	A new record in both tables of the database was created	✓

Repeated for all pages in the top 100. Additional tests for other possibilities, such as books in the database and in the top 100 and books in the database but not in the top 100, can be found in the testing section of the appendix.

### **6.3.4: updateBook**

Updates a book in the database. Assumes that the ASIN passed in is correct from validation in the UI, and uses the same code as addBook, specifically the iteration through the list in the HTML to extract data, so testing required was minimal.

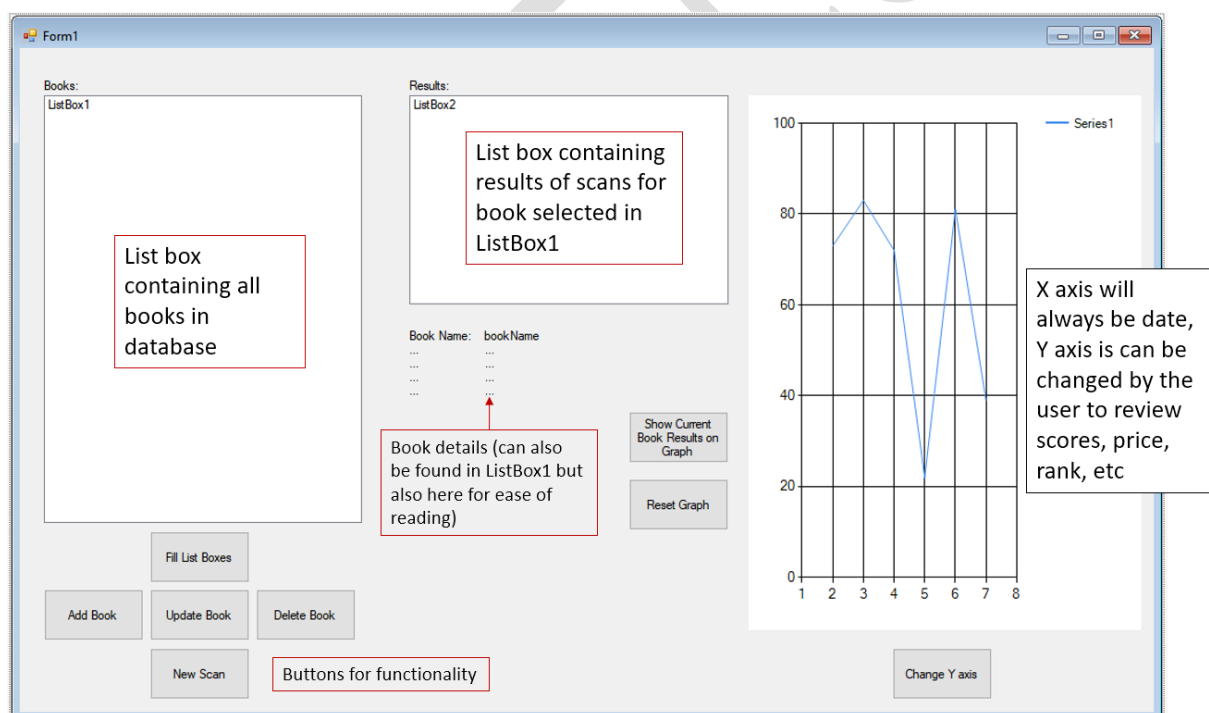
Test	Input Data	Result	Expected result?
1	<p><b>B004T0A30A</b></p> <ul style="list-style-type: none"> <li>Book in the database, values for Length, File Size, Language and Publisher were null.</li> </ul>	Book was updated with new information	✓

## 6.4: GUI

A major issue was encountered during the development of the GUI for the system.

Originally the entire program was meant to be written in Python with the use of TkInter or PyQt5. After some time spent attempting to learn UI design in Python using a purely text based IDE it was decided that it would be infeasible to produce a suitable GUI in the time given with no previous experience.

To solve this problem a GUI was instead developed using a more graphical IDE and language- Visual Studio and Visual Basic. This made it much easier to develop a satisfactory UI in the time given. The first design of the UI was sent to the project supervisor and deemed to be acceptable for the system.



This was not the final design of the GUI. The final design can be found in the design section of this document.

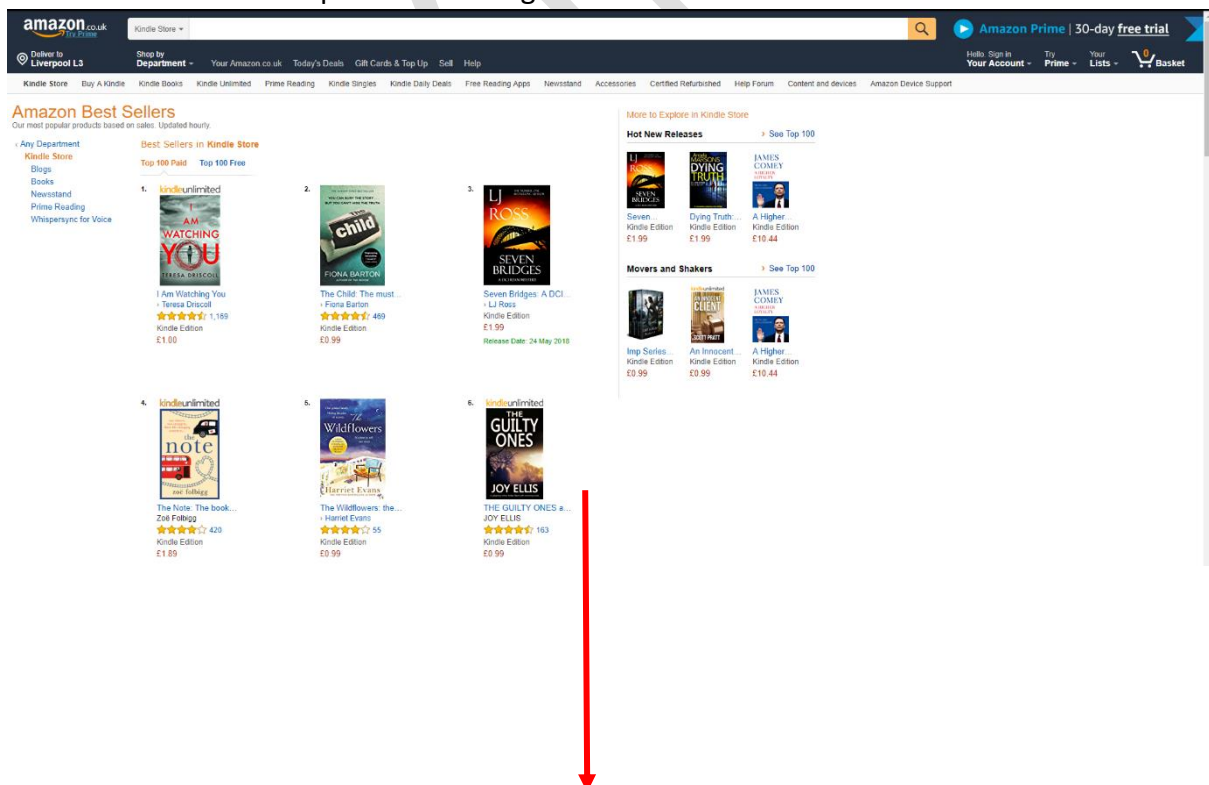
Most errors found during this stage of development were syntactical in nature and will not be included in this document.

Since a large amount of the code associated with this part of the system is manipulating the various Windows Form objects available in Visual Studio, and the rest of the code is simply

calling the already tested Python files, testing of the UI will not be included in this document and instead can be found in the testing section of the appendix.

## 6.5: HTML Change

Approximately 1 week before the demonstration of the system was to be given, the HTML of the Amazon Kindle top 100 was changed.





- The top 100 list was now spread over 2 pages instead of 5
- The top 100 list now contained 50 books per page rather than 20
- Some of the locations of data in the HTML had changed

Tools to scrape the new version of the Amazon Kindle top 100 were produced in this week but due to time constraints were not fully tested. During this week the webpage would be changed back and forth between its old and new design, resulting in one version of the program working one day and not working the next. The webpage could also change design during the run of the program due to utilising the same URLs- the program could scrape the first two pages normally through the old format and the next webpage would be in the new format, which the currently executing code would not work on. Due to the unstable nature of the webpage it was decided for the purpose of the demonstration that the old version of the top 100 would be saved locally and scraped.

## 6.6: Prototype 3.5.0 (soupTest3Selenium)

This version of the program was produced with Selenium in an attempt to see which version of the webpage was being accessed during the scan and allowed the programmer to stop the code from executing if the design of the webpage was changed during the program run.

Code is identical to Prototype 3 aside from the webpage being accessed in a different manner.

## 6.7: Prototype 3.5.1 (soupSelenium)

This version of the program only accesses two pages of the Amazon Kindle top 100 and scrapes all 100 results from them. The code has been altered from prototype 3.5.0 to work with the new HTML and to scrape fifty results from each page over two pages. Like Prototype 3.5.0, utilises Selenium to ensure the correct pages are being scraped.

## **6.8: Prototype 4**

Contains all the code for the top 100 scrape on new pages without selenium. Not too much code has changed from Prototype 3.5.1.

Since the webpage was constantly changing it was decided that it was pointless to try to create a final version of the program, utilising the Visual Basic UI, with the code from this prototype. If the system were to continue in the future and Amazon were to keep their new design, code from this prototype would have to be split in the same manner as the separated Python files used in the demonstration.

## **6.9: Demonstration Build (Final Build)**

The final version of the project was produced for the demonstration. This version scrapes locally hosted webpages whose layout is known, rather than scraping the, as of now, unstable live webpages.

To create this version of the system the code from Prototype 3 was split into a number of separate files, a method to call each of these files and display the results of the called Python code, was written for each button on the GUI.

Additional code was created for other elements of the GUI, such as labels being updated whenever a new book is selected, which was relatively simple, with most errors being syntactical and not significant enough to include here.

One problem worth mentioning here was when some feedback was to be passed from the Python code to the Visual Basic code. More specifically the Python code was intended to return some text giving the status of the code being executed, such as which page in the top 100 is currently being scraped, or any warnings thrown as a result of SQL queries. These were intended to be shown in a text box on the form itself but attempting to do this was unsuccessful. Due to time constraints this problem was not solved. Instead the program now simply shows the Windows console and the Python code outputs data directly to the console instead. Details on this problem can be found in the testing section of the appendix.

## **7.0: Evaluation**

Evaluation was performed throughout the project through meetings between the creator of the system and the project supervisor. These meetings were mostly informal and generally consisted of a small demonstration of what had been produced/found since the last meeting, followed by some feedback being given by the project supervisor which was used to improve the project or clear up any misconceptions. Transcripts of the meetings were not kept. Initially questionnaires were to be produced and filled out during each meeting, but this was decided against and a more informal discussion of the system occurred.

The project as a whole was evaluated by comparing the essential and desirable features from the specification against the final program, and also by using the evaluation design found in section 5.11 of this document. The system would be deemed a success if all essential functionality was present. Any desirable functionality present is a bonus. The creator of the system was involved in the evaluation of essential and desirable functionality, whilst feedback from the project supervisor was used to get a greater idea of how the program could be improved.

### **7.1: Essential Features**

- Allow the user to scrape the Amazon kindle top 100 and extract details

The system successfully allows this. This essential functionality is present.

- Allow the user to scrape the Amazon top 100 to find a specific book, given its URL

The system successfully allows this. An ASIN is used instead of a URL but this can be easily edited and each can be derived from the other.

- Store basic information, URL and current rank of each item in a database

The system stores an acceptable amount of information about each item in the database.

- Show the user an overview of a specific item such as; how long it has been in the top 100; how long it has been at its current rank; the item's highest historical rank; and dates/times of when data was retrieved.

The system provides an overview of specific items in the database.

- Provide a simple interface to allow the user to perform the above

A UI was produced in Visual Basic which all essential functionality can be accessed through.

Since all essential functionality is present the project can be deemed to be a success.

## **7.2: Desirable Features**

Any desirable features present in the final version of the system contribute to further success of the project.

- Construct a graph, shown to the user upon request, which shows how the ranking of a book has changed over time

A graph was implemented in the UI of the program and shows how a book's rank has changed over time. The graph also allows the user to see how other pieces of information about each book has changed over time too, these being the review score, the number of reviews and the price of the books. This desirable feature is fully implemented

- Allow side by side comparison of at least 2 books

The graph produced allows multiple books to be plotted upon it at the same time, which the user can then use to see some comparison. This desirable feature is present.

- Allow the user to scan both the free and paid top 100

This desirable functionality was not present but could be implemented with relative ease by changing the passed in URL when scraping.

- Automate the process of scraping each hour for as long as the user is running the program

This desirable functionality is present in the final version of the program through the use of a timer.

Most desirable functionality as laid out has been met. Therefore it can be deemed that this project was, overall, a success.

### **7.3: Evaluation from Design**

From evaluation design in section 5.11 of this document:

- Does the system successfully scrape the Amazon kindle top 100 list? – Yes.
- Does the system successfully find details of a book given the correct URL? – Yes.
- Does the system store all the necessary information for each book? – Yes.
- Does the system provide sufficient analysis on any given book?
  - Does the system show how long a book has been in the top 100? – The system shows the first and last time a book has been found in the top 100 and also provides a graphical overview of each time the book was found.
  - Does the system show how long a book has been at its current rank? – The graph in the GUI of the system can show this.
  - Does the system show the book's highest historical rank (since the first time it was found in a scan)? – The graph in the GUI of the system can show this.
  - Does the system provide dates/times for specific scans? – The graph in the GUI of the system can show this. The GUI also provides a number of labels to show the values of each field of a result in the results table of the database, one of these being the date of a scan and another being the time of a scan.
  - Does the system provide functionality to see results of a specific scan on a book? – The GUI provides a number of labels to show the values of each field of a result in the results table of the database.
- Does the system have an adequate UI? – For the purpose of the demonstration the system's UI was adequate. The GUI can greatly be improved in future versions of the system to provide greater analysis capability and enhanced usability.

Some additional feedback was received shortly after the demonstration of the system had taken place. It was briefly discussed how the GUI could be improved to provide further analysis on the results by using multiple graphs, so the user can spot trends. Another possible feature discussed was keeping specific track of flash sales of a book and seeing if other data changed as a result. This is currently partially implemented in the program, as the user can see when a price drops and then check other data from there, but the single graph makes it difficult to see trends.

### **7.4: Results/Outcomes**

The resulting program from this project does not work on a live version of the Amazon Kindle top 100. As of the time of finishing this project, it is infeasible to create a system which will work on a live version of the top 100 due to the changing nature of the webpages. This is out of the control of the project and solely as a result of unfortunate timing. If the project were to continue one would have to wait until a final design is settled on by the creators of the Amazon Kindle top 100 webpages or create a more robust system capable of detecting which page has been accessed and changing the scraping loops



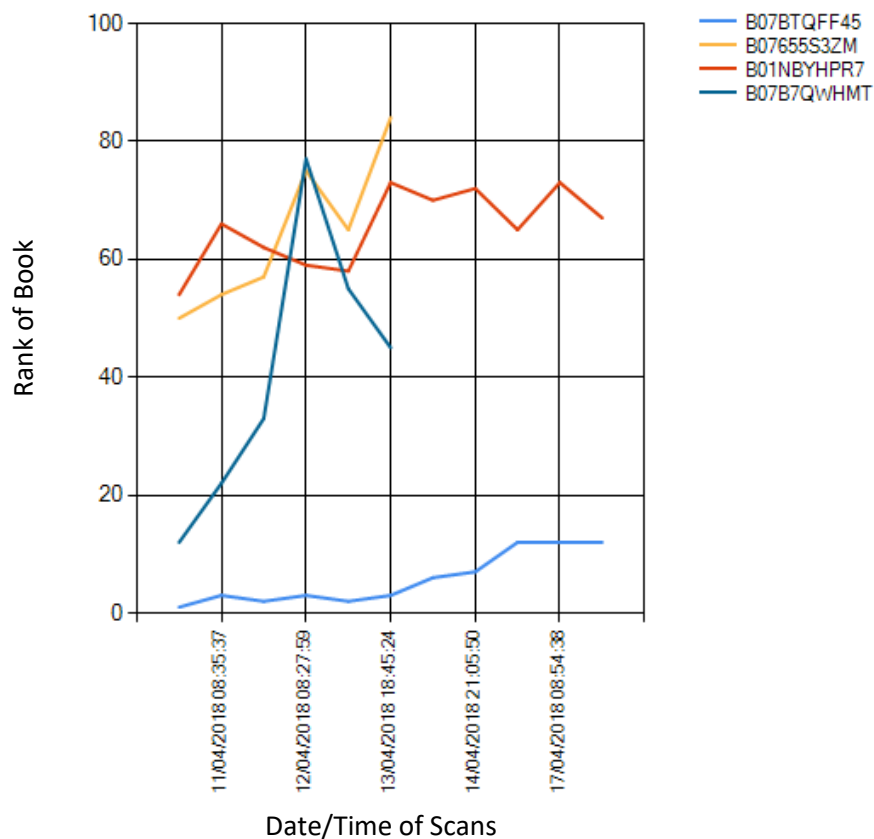
accordingly. Given more time than one week, such a tool could possibly have been produced.

It could also be said that not enough analysis has been performed on the results of the scrapes at this time. The purpose of the final build of this project was to demonstrate the benefits of the approach taken and the system does give a good idea of what analysis can be performed on the longitudinal data scraped. In the event of the continuation of the project more advanced analysis could be performed on the same data which is scraped by this version of the system, such as looking for an author that appears more often than any other or calculating the average time that a book tends to stay in the top 100 for. This analysis would have to be performed on data gathered over a much longer period of time to be of any real interest, so implementation of such analysis was not included in the final version of the project due to time concerns.

Even something as simple as adding additional graphs to the UI could also be realised to allow the user to view the data more meaningfully and spot trends by eye.

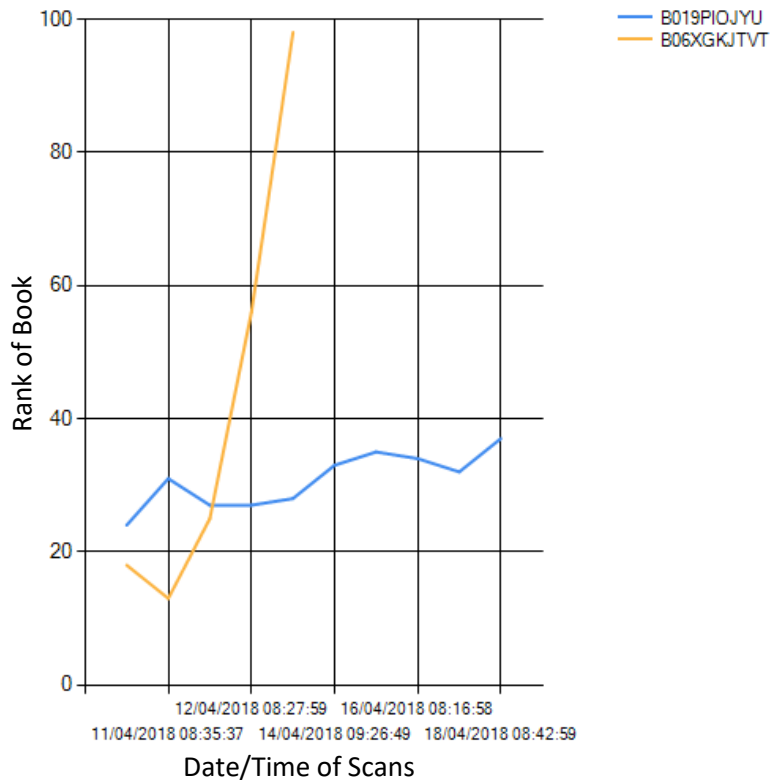
## **7.5: Results**

Below are some example trends that were found with the current version of the system.



Even over a short time period it is interesting to see how the rankings of books change over time. It appears that some books seem to hold around their position for extended periods of time, such as B07BTQFF45 remaining in the top 20 for over a week and B01NBYHPR7 remaining around rank 65, while other books tend to fluctuate wildly before dropping off altogether as in the case of B07B7QWHMT. It would be interesting to see if these trends continue but, due to the HTML for the Amazon Kindle top 100 remaining unstable, as of now it is infeasible to continue collecting data with the current version of the system.

It also seems there are definitely favourites in the Kindle top 100.



In this graph B019PIOJYU is a book originally released in 2001, while B06XGKJTVT is a newer release that started off strong but very quickly dropped off. It will be interesting to see in the future if older, classic books tend to dominate the Kindle top 100 while newer books fade quickly.

## **7.6: Strengths and Weaknesses**

### **7.6.1: Strengths**

- Meets all essential functionality
- Meets most desirable functionality
- UI developed is sufficient
- Code is split into multiple files so it is easy to update specific parts of the system

### **7.6.2: Weaknesses**

- The system may not work on a live version of the Amazon Kindle top 100
- The system may require continuous upkeep if the design of the webpages are changed further
- More validation is required on the system. Since only myself used the system I was able to make sure I did not input anything unexpected but for any future versions of the system that will be used by different people some validation on inputs will be required to ensure the program does not break.

## **8.0: Learning Points**

Overall I believe that the project went well. Creating a program which would interact with a live webpage was something I had not done prior to the project and proved to be an interesting experience. I had also never coded with Python before the start of the project so learning to use a new language under a time constraint was an enjoyable challenge.

No major difficulties arose during the creation of the program until around a week before the demonstration was due to be given. The layout of the webpages I was scraping for the project were completely changed and my program no longer worked. I managed to create a working version of the program for each of the different layouts of the webpages but as a result of the obstacles that arose due to the now unpredictable nature of the webpages and resulting unexpected time loss I was unable to include as much analysis of the data gained as I wanted into the demonstration build.

Admittedly this could have been less of a problem if I had put more thought into time management. Naturally the contents and layouts of webpages are subject to change at any point and it was optimistic of me to think that the webpages would stay as they were throughout the entire project. I feel this can partially be forgiven since the layout of the webpages had not undergone major changes in over five years, but I do need to be more thorough during future risk analyses.

Ultimately, I was still able to complete all of the essential features present in the specification document along with the most time consuming of the desirable features (this being the graph showing how a specific item's stats have changed over time). The rest of the desirable features could quickly be implemented now that the groundwork is laid, along with any additional features for trend analysis and UI improvement.

UI design is what I believe to be my weakest skill in software development and I initially wanted to improve this by using some new UI development tools, namely TkInter or PyQt5. After attempting to create an interface with these text based tools I feel my ability to design UIs has improved despite not actually using these tools in the final build of the project. Due to timing issues I decided to use a system more familiar to myself (Visual Studio) and its simplified drag and drop interface building feature, but I learned a lot from attempting to branch out.

I feel as a whole that the project was a success and fantastic learning experience. Creating a system spread over multiple languages was a new experience for me and I found it to be both enjoyable and rewarding. Keeping a record of a large amount of changes and complications that occurred during the project was and is, now looking back, a great way to see how even a project that has been carefully designed will still likely undergo many changes even after the design stage.

Building a program on behalf of a client was also an interesting experience. Presenting my work to the client (project supervisor), discussing the project during meetings and meeting deadlines has helped prepare for similar situations which are bound to occur during future

software development tasks. I feel more confident than ever giving presentations as a direct result of this project.

The project has given me a great insight into working on a larger system over an extended period of time and as such has helped immensely in preparing me to enter the software development industry.

## **9.0: Code of Practice**

This project was completed in accordance with the British Computer Society's Code of Practice [13].

Referring to Section 3 "Key IT Practices":

- The 'customer', in this case the project supervisor, was worked with in order to reach a common understanding regarding deliverables and the project structure
- When planning the project the scope and deliverables were agreed in advance in the form of a specification document
- Similar projects were researched to learn more about the project area
- Realistic estimates of the timescales for the project were made
- When performing a risk assessment some risks were identified. The risk which actually did occur during the project (the HTML of the webpages changing) was not identified during the risk assessment but should have been
- Metrics were kept on project activities
- Mistakes made were honestly summarised and lessons were learned

Referring to Section 4.2 "Research":

- Unauthorised research on human subjects did not occur
- No research on animals occurred
- No private data was used during the project
- Other research was investigated, and their contribution acknowledged

Referring to Section 5.2 "Software Development":

- Well structured code allowing testing and maintenance was produced
- Language appropriate guidelines were followed, in the form of official documentation and other online resources
- Meaningful naming conventions were used
- Code produced was in line with the design specification
- Test cases were planned to cover many routes through the program
- Values from test were checked to be as expected in the database as well as on the program UI
- A testing log was kept
- A log of anomalies found during testing was kept and details on how they were resolved were recorded

The project was also conducted in accordance with the British Computer Society's Code of Conduct [14].

Referring to section 2 "Professional Competence and Integrity", I made it clear early in the project that I had never used Python before so did not claim any level of competence not possessed. Alternative viewpoints, in the form of other projects and feedback from the project supervisor and second marker, were valued during the project.

DO NOT COPY

## **10.0: Bibliography**

1. Amazon.co.uk Bestsellers: The most popular items in Kindle Store [Internet]. [cited 2018 May 4]. Available from: <https://web.archive.org/web/20130310014828/https://www.amazon.co.uk/Best-Sellers-Kindle-Store/zgbs/digital-text>
2. Glez-Peña D, Lourenço A, López-Fernández H, Reboiro-Jato M, Fdez-Riverola F. Web scraping technologies in an API world. *Brief Bioinform*. 2014 Sep 1;15(5):788–97.
3. nicerobot. Home [Internet]. Web Scraping Service. [cited 2018 May 5]. Available from: <https://webrobots.io/>
4. Web Robots. Instant Data Scraper - YellowPages.com demo [Internet]. [cited 2018 May 5]. Available from: <https://www.youtube.com/watch?v=biHNChKt0mA>
5. SysNucleus. WebHarvy Web Scraper [Internet]. [cited 2018 May 9]. Available from: <https://www.webharvy.com/>
6. sysnucleus. Scraping Amazon product title, ASIN, BSR, rating, reviews, weight from list of URLs using WebHarvy [Internet]. [cited 2018 May 5]. Available from: <https://www.youtube.com/watch?v=HCBVrgMc4s8>
7. Bonifacio C, Barchyn TE, Hugenholtz CH, Kienzle SW. CCDST: A free Canadian climate data scraping tool. *Computers & Geosciences*. 2015 Feb 1;75:13–6.
8. Beautiful Soup Documentation — Beautiful Soup 4.4.0 documentation [Internet]. [cited 2018 May 9]. Available from: <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>
9. Ray S. Beginner's guide to Web Scraping in Python (using BeautifulSoup) [Internet]. Analytics Vidhya. 2015 [cited 2018 May 9]. Available from: <https://www.analyticsvidhya.com/blog/2015/10/beginner-guide-web-scraping-beautiful-soup-python/>
10. Yek J. How to scrape websites with Python and BeautifulSoup [Internet]. freeCodeCamp. 2017 [cited 2018 May 9]. Available from: <https://medium.freecodecamp.org/how-to-scrape-websites-with-python-and-beautifulsoup-5946935d93fe>
11. Human Subjects Research Definition [Internet]. RDIA. 2012 [cited 2018 May 9]. Available from: <http://oria.gmu.edu/research-with-humans-or-animals/institutional-review-board/human-subjects-policies-procedures-forms-and-instructions/human-subjects-research-definition/>
12. Welcome to MySQLdb's documentation! — MySQLdb 1.2.4b4 documentation [Internet]. [cited 2018 May 9]. Available from: <https://mysqlclient.readthedocs.io/>
13. cop.pdf [Internet]. [cited 2018 May 9]. Available from: <https://www.bcs.org/upload/pdf/cop.pdf>

14. conduct.pdf [Internet]. [cited 2018 May 9]. Available from:  
<https://www.bcs.org/upload/pdf/conduct.pdf>

DO NOT COPY